

SyncWare News

The journal for technical
applications of T/S
computers

Volume 2 Number 5

May - Jun. '85

SyncWare Meets QL

Sinclair Research has been promising the North American arrival of their new QL computer for so long, it's hard to remember just when the first mention of it was made. Were it not for the fact that this machine is being sold in the U.K., one might well wonder if it's just the figment of someone's fertile imagination.

Like many fellow Timex/Sinclair fans, we at SyncWare were anxiously waiting for that first boat load to reach our shores, but not long ago we decided to cheat a little. We sent one of our "international correspondents" off to merry olde England with a bag-full of travelers checks. He returned with one very real QL computer.

So starting with this issue, we can give you accurate details of just what this computer is about. We'll tell you why you might (or might not) want to buy one of your very own. And if you decide that the QL is for you, the SyncWare News ace technical staff will at least try to get you through the "OK, I got the cursor blinking... Now what?" syndrome.

U.S. & U.K. Versions Different

Before we go a step further, one very important point must be mentioned. The computer we received is an English computer. It never was intended to be used here. North American versions will be different in several hardware and software aspects, but we have been assured by Mary Reinman, spokesperson for Sinclair Research in Boston, that functionally, the U.K. and N.A. versions are identical. Besides the common power supply and video problems which crop up (in the U.K. they use 220v @ 50hz AC) in our English computer, the joystick and serial RS232 port connections on the back of the computer are very non-standard. In addition, the input/output device addresses will, in all likelihood, be different from the American version. Mary Reinman states that the QL's they sell here will be suitably "Americanized," and connectors on the back will be standard.

The Owner's Manual: What it Is...

We didn't even have to get our QL running to realize that this is no ordinary computer. Our first clue was the owner's manual. It is a 400 plus page book in a full size looseleaf binder. The manual is a concise guide to hooking the unit up, and running it. One of the larger sections of the book, titled "QL Beginner's Guide," delves into programming concepts; e.g., making the computer do what you want it to. In addition, this chapter offers a rich introduction to the BASIC functions and commands at your disposal. Many short, easy to enter examples are provided.

Another very impressive chapter lists every command and function that the QL understands. In a format designed to serve as a handy and effective reference guide, it tells you what happens when you execute the given command, the proper syntax and several examples of how the command can be used.

...And What it is NOT

Is the manual complete, easy to understand, and will it teach you how to program? No, but you shouldn't expect it to. It was written to show you what the computer is capable of doing, and this it does admirably. The owner's manual is, in fact, incomplete in spite of its 400 odd pages, it is very difficult to understand, and it will not turn you into a programmer. Don't let this discourage you, however. A computer is a TOOL which you operate by communicating with it via a written LANGUAGE. To draw an analogy, the directions which come with a power saw won't turn you into a carpenter. An English text book won't turn you into a writer. Only a great deal of study and practice will do that. It's the same with computers.

First Impressions of the Hardware

By all standards, the QL's "nuts and bolts" hardware is a dramatic improvement over the other ZX/TS computers. First of all, you get a real live typewriter keyboard with genuine push keys. A typist would find the keyboard quite acceptable for word processing. At the rear of the machine are the connections to the power supply, 2 RS-232 serial ports, TV or monitor output, two joystick ports, and a provision for cartridge software. Another expansion port is brought out to the left of the keyboard. Every pin of the 68008 CPU is accessible from this connection, and it is here that additional peripherals as well as the rumored Sinclair 1/2 Megabyte Rampack will be attached.

Microdrive Mass Storage

Built in to the case on the opposite end of the keyboard sits a pair of those famous (infamous?) micro-drives. Each holds a matchbook size "endless loop" wafer which is capable of storing 100K of data. Up to 6 additional drives can be attached.

Ever since the TS2068 came out, the microdrives were hyped as the end-all answer to slow and tedious cassette loading of programs. So far, SyncWare's experience with these new devices has left something to be desired. It is true that they are fast. A complete pass of the tape can be made in a little over 7 seconds. This means that in theory at least 100k could be loaded in a very short period of time. The QDOS operating system which handles these drives (among other things) is the epitome of efficiency and good design. However, like the Sinclair 16K rampack for the ZX81, we found the microdrives cannabilistic in the way they treated programs and data. In approximately 30 hours of use, the microdrives managed to unravel 2 of our wafers. In addition, one of the bundled programs which came with the computer loaded only once before it died. Now we will admit that perhaps our failures were caused by our own bungling. Hopefully, we will learn the error of our ways. But we are disheartened by their initial performance nonetheless. A good fast disk interface coupled with the flexibility of QDOS would make a world of difference.

Inside the case, there's 48K of ROM which contains the Sinclair SUPER BASIC and the QDOS Operating system. 16K more ROM can be plugged into the back of the computer. For RAM, there is 128K built-in. Of this, 32K is set aside for the video display. The remaining 96K is used for Basic and/or machine code programs, system variables and In/Out to peripheral devices.

As previously mentioned, the QL uses the Motorola 68008 microprocessor chip. This is a very advanced chip compared to the Z80A, which is the mainstay of the ZX81/TS1000 and Spectrum/2068 machines. Up to a megabyte of directly addressable memory can be handled by the 68008. The QL uses a more sedate 8049 co-processor to handle all Keyboard and sound functions, thereby freeing the main processor of these housekeeping chores. The result is a faster running computer.

All in all, the QL's hardware is quite remarkable. Even with the microdrive problems we remain impressed. Sinclair has a history of improving upon itself. We see no reason to believe any different now.

This article has only skimmed the surface of just the hardware half of the amazing QL computer. Next time we'll tell you about the computer's programmability which is every bit as powerful as its circuitry. Stay tuned...



FOR YOUR SUPPORT

This column announces any software or hardware that is new or otherwise untested by us. If you have something of interest, send us a description of your product. We advise readers to send a SASE, as a courtesy, to the individuals or companies, for further information.

C.W. Associates, 419 N. Johnson St., Ada, OH 45810, has two 2068 programs available. Supershaper is a graphics utility program and City 2068 is an adventure type game. Write for info. Prices were not included with the documentation.

Gulf Micro Electronics, 1317 Stratford Ave., Panama City, FL 32404, introduces Smart Text ZX/TS, for the 1000/1500. It is a word processor/dbase/interactive office utility program. It will work with the JLO Video upgrade and an assortment of printer interfaces. \$29.95.

Aerco, PO Box 18093, Austin, TX 78760, (512) 451-5874 has their disk interface ready now. It is 2068, Spectrum and CPM compatible. It comes with an RGB output and 64K RAM in the cartridge bank. \$199.00 for the FD-68. Call for group discounts. A complete system is available for \$400.00.

G. Russell Electronics, RD1 Box 539, Centre Hall, PA 16828, has the ROMFIX available as a mate to your Romswitch (reported in SWN2/3 Forum) for installation under the Exrom. This allows running of almost all Spectrum programs on the 2068 (with Spectrum ROM). \$3.25 ppd.

Melvin MacKaron, PO Box 14466, Albuquerque, NM 87191, (505) 884-8391, has an electronic gradebook, The Complete Teacher, for the 2068 and other computers. Write for info. No prices were given.

Ace Software, 2 E. Oak Ave., Moorestown, NJ 08057, has two budgeting programs available for both computers. Payoff will keep track of what you owe on your credit cards and for how long and Payout is for your monthly forecasting. They are \$14.95 each.

Rheesware, 1660 S. Duneville, Las Vegas, NV 89102, has both serial (\$59.95) and parallel (\$69.95) printer interfaces that plug into the 2068 joystick ports. They also have an analog adaptor which allows hookup of analog joysticks, mouse or touch tablet. \$54.95

EZ-Key, Suite 75, 711 Southern Artery, Quincy, MA 02169, (617) 773-1187, has a keyboard interface for both computers. It plugs into the edge connector so that you do not have to open your computer to plug a keyboard in. KBD-1 for the 1000 and KBD-2 for the 2068. \$39.95 + \$2.95s&h each.

Falmouth Computer Service, 255 Falmouth Rd., Falmouth, ME 04105, (207) 781-4877, has an amendment to Vu-File (2068) which allows single tape loading. \$8.00. They also have 16K RAM packs for \$15.00. Converted for the 2068, \$35.00. Add \$2.00s&h

Damco Enterprises, 67 Bradley Ct., Fall River, MA 02720, (617) 678-2110, has the Rotronics Wafadrive. It comes with a Spectrum Emulator, parallel and RS232 ports and 2-128K drives. The whole sytem is \$229.95. Add \$5.00s&h

A.F.R. Software, 1605 Penna. Ave., #204, Miami Beach, FL 33139, (305) 531-6464, has T/S-Text 2000 for the 2068. This word processor allows editing of 650 lines of text in 32 columns for the 2040 printer. \$19.95

John Oliger, 11601 Whidbey Dr., Cumberland, IN 46229, has 2068 cartridge boards, which gives the eproms that you burn your favorite programs on a place to live. \$11.95 bare, \$15.95 with parts, the expansion board is \$14.95 bare, \$43.95 with parts (has RGB output capability). Also, the 4.4 Volt/22 Volt power supply board (necessary for the eprom programmer) is \$4.95 bare and \$9.95 with the board parts. You supply the transformer and fuse.

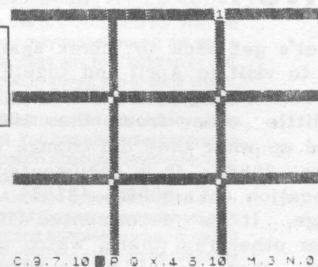
Van Vangor, Bethlehem Tool, Box 346C Retreat Rd., Island Falls, ME 04747, (207) 463-2835, has offered to sell the Plug Guard, see text and drawing on page 44 of SWN Vol. 1 Reprint, for \$7.00ppd.

T-Ware, 40 Aspen, Great Falls, MT 59405, (406) 452-5673, has two educational games available for the 1000. Mr. Math is a compiled, graphic tutor (ages 5 and up) and Spell World is a spelling driller. \$10.00 each or \$12.00 each on the Aerco disk.

Integrated Data Systems, 30 Brookmount Rd., Toronto, M4L 3N1, has a 2068 emulator board for \$10.00, Spectrum ROM for \$22.50. They also have for both machines an Eprom burner board (\$32.00), power supply (\$25.00) and reader board for \$15.00. Add \$1.50s&h

TIMESCREEN™
ZX81, 16K

Creativity and
planning aid:



- 3 screen formats (9-block format shown here)
- BASIC routines to enter and rearrange data
- Calendar
- 10 screens in 16K; 36/32K
- Use with any printer that will accept COPY routine

Cassette and manual, \$9.00 (VA res. add 4% tax), + \$1.00 s&h for one, 50¢ s&h ea. addl. cassette. One 8½x11 Label-Index Card sent with each cass.

HAWC™
Programming
4604 Apple Tree Drive
Alexandria, VA 22310

FORUM

Oh Yes... Oh NO!!

It's apparently true that Timex Portugal is going to bring back the TS 2068 as the Timex 2068. They have the 3" floppy drives for it too. That's just grand. Unfortunately, it will have a Spectrum backplane which may make them incompatible with some of the US TS 2068 peripherals. It will come with a Spectrum ROM cartridge though. I'm glad that Timex is such a considerate company. (The disk drives will be available for both types of 2068's.) Will the Timex disk I/F be 2068, Spectrum AND CPM compatible?

They are keeping their buggy ROM and useless joysticks and giving up the 1000 compatible expansion port. How about a couple of Kempston compatible joystick ports built in. Come on guys, let's get creative!

Timex has a tremendous public relations task at hand. We do want to see 2068 computers brought back, but not in bits and pieces. I pose a question to all of you.

Would you prefer to see Timex sell the 2068, or would you like to see Sir Clive get the North American rights back and come in with the Spectrum+? Do we need another half-breed unsupported computer in this country? I'm sorry but that's the way I feel.

Oh No... Oh YES!!

Well, let's get back to Timex again. Fred Nachbaur came to visit in April and together we discovered a VERY serious bug in the TS1500. They (Timex) changed very little code from the 1000, but they really screwed up what they did change (probably practicing for the 2068). It is documented that on power up, the location at 2000h (8192) is tested for a cartridge. It is decremented (if you use a Hunter board or other ram there, watch out). If the result is zero, then you JUMP to 2000h and proceed. That was bad enough! In the course of writing that inane bit of code, they screwed up the LOAD routine and they DID NOT fix it. On a bad load, you will jump to the byte after LD HL, RAMTOP (into the middle of it) in the NEW routine. Consequently, the stack pointer will be placed at the memory location where the bad load occurred. Ramtop will not appear to change. If you are lucky, when the screen clears you will return to 0000h and start fresh. If you are not (e.g., you loaded almost to the end of the program), then on the next load you will crash again. We will have the full report for this and all those other "bugs" in the next issue, when we give you lots of things to do with your eprom programmer.

Oh Well

Dear Editor, I just received 2/4. It looks great, but you forgot a couple of things and made a few mistakes.

1. You didn't mention that an expansion board is needed on the 2068 for the programmer, but say that it is needed on the 1000(?)

2. You referred to that extra gate (decoding the EXROM) last time saying it would be used in the next issue, but you left it out along with the 2068 ROM changes.

3. And those pullup resistors! I'm glad you didn't put my name on that one. You are wrong! You added 7 pullups when 8 are required. (Don't you like D2?) This will find its way into other publications and everybody will be wrong. The pullups supply the low order address for IM2 (there is a lot of software like this) driven software.... The keyboard is not one of the things that these help. Also the issue 1 Spectrum put an FFh on the bus, not the Issue 3 (puts out a BFh)

John Oliver

Dear John, I stand corrected on the keyboard point (but now you know why I bend your ear on the phone so much). Even worse than splitting the programmer article into two, the Cassette Connection got bumped. I referred to that in the first paragraph on page 1. If you solder an edge connector on the programmer, then you do not need an expansion board (although it is preferred). You need a mother board on the 1000 due to the width of the board. The 2068 schematic shows D2 with a 10K pullup on it, just right of U16. Well, I retract your name from the list of associated people. I promise to take better notes next time. (See Support for more information.)

Pro/File 2068 Books are Shipped

Tom woods informs me that all Profile 2068 manuals have been shipped. If you didn't get one or you would like to get one, then get in touch with him directly.

Q/A

Dear Editor, I am writing to enlist your help. I recently bought an A and J Microdisk. I like the unit well enough, but I have run into a small problem. I have Bill Russell's Romswitch and found that I cannot SAVE or LOAD Spectrum programs to the Microdisk. Can you shed any light on this?

Ken Duda
210 Bernice Drive
Northlake, IL 60164
312 562 5898

You have run into a big problem. The A&J system uses the EXROM for various purposes. The Spectrum part of the Romswitch will work even if you plug the EXROM in backwards. I won't even mention that the routines in each of the ROMs are in different locations. It will be necessary to copy the operating system into RAM and convert all the addresses that make specific ROM calls. It may require relocating the the A&J code to a high memory location and let it reside there. It's a big job! If anyone made this change or is contemplating it, please get in touch with Ken. He'd like to talk to you.

Spectrumizing

Spectrum ROMs are available from many sources. There are three methods for installation. You can plug in a ROM directly, use a cartridge board or install a switch unit. Check with your nearest dealer or get in touch with the following people:

Romswitch- G. Russell Electronics, PO Box 539, Centre Hall, PA 16828, (814) 364-1325

Cartridge board- Doug Dewey, 206 James St., Carrboro, NC 27510

ROM- English Micro Connection, 15 Kilburn Ct., Newport, RI 02840, (401) 849-3805
Write for more information.

BUG ALERT!



The last three bytes were omitted from the decimal listing of "CLEAR THAT SCREEN" in Vol. 2:4. These bytes are: 245, 008, 201.

VOLUME 1 book, page 62: In Figure 2b, pin 3 of the top LM311 should be labelled "Sync Ref." (reference), not "Sync Ret." Also, the resistor from ULA pin 16 to ground was not labelled; this part is 10Kohm.

VOLUME 1, Page 91: The last paragraph was chopped off in the middle; it should have been chopped entirely. Page 94, column 2: The least square equation for a linear fit should have the quantity on the right $(Y(\text{calc.}) - Y(i))^2$ SQUARED. Similarly, the summations in the equations in the following paragraph should be squared.

Volume 1, Page 96:

$$F''(X) = -\text{SINE}(X/57.29)/57.29^{**2}$$

$$F'''(X) = -\text{COS}(X/57.29)/57.29^{**3}$$

$$F''''(X) = \text{SINE}(X/57.29)/57.29^{**4}$$

Busted Chips

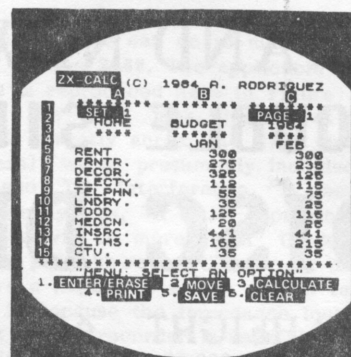
It has come to my attention that Timex no longer has parts for the 1000 computers, although they will fix your computer for a fee. Sinclair Research, 50 Staniford St., Boston, MA 02114, will supply ZX-81 parts. There are other suppliers for these parts (see Support 2/3, 2/4) also.

In a conversation with a knowledgeable individual at Timex, it was pointed out that the 2068 SCLD chip is no longer being made (NCR made it originally). In other words, when it goes, so goes your computer. However, it was also pointed out that it is difficult to "blow it" unless you really try. One problem that has come up is that the modem hook up may give you grief. This problem occurs when switching power supplies on and off in the wrong order. Although your computer may go down, it is not out. Since you can no longer send it back to Timex for repair, replace the two 74LS245 buffer chips (about \$4.00) and get back on line again.

Since 2068 computers are being made in Portugal and sold in Europe, I'm pretty sure that the SCLD chip is still being made. The source for these should not dry up too quickly. I'm glad that we can always get straight answers from "Mother Time."

POWERFUL AND INEXPENSIVE BUSINESS SOFTWARE FOR ZX81, T/S1000 and T/S1500 COMPUTERS

ZX-CALC



An electronic spreadsheet calculator is the fundamental basic tool for summarising, reporting and analyzing in matrix form any accounting, mathematical or scientific manipulation of numbers. ZX-Calc operates in 32-64K RAM and affords a maximum of 3360 characters / spreadsheet. The entire matrix consists of 15 columns (letters A-O) and 30 rows (numbers 1-30) with 8 characters / cell. Unlike other popular ESCs, ZX-Calc uses in calculations and within cells all 14 math functions on the ZX-81 / TS1000. It offers a unique "SUM" function that totals one or more rows / columns simultaneously. Parenthesis can be used within equations. There is no fixed limit on how many equations may be entered. Formulas may be stored in all 420 cells of the spreadsheet. The display affords 15 rows / columns. Loading of data into more than one cell can occur across / down one or more row / column simultaneously. With vertical windowing you can arrange a set of columns in any order, or practice using fixed-variable-alignment display formats. The menu offers 6 options: enter / erase, move, calculate, print, save and clear the spreadsheet. Enter / erase allows the entering, deletion or data alignment within a cell through the use of a mobile cursor. With the move option you may move around the entire spreadsheet to access any row, column or cell. The calculate option allows you to enter labels, values or formulas into a cell or write and enter equations that will act upon the data already within the spreadsheet. You can also enter bar graphs into a cell in this option. Absolute / relative replication, down / across a column / row, is also allowed by this option. Also this option allows the automatic calculation of the entire spreadsheet with one single command. Print allows you to output to either the ZX / TS printer the entire spreadsheet by column-sets and row-pages through use of the COPY command. The entire spreadsheet may be saved on cassette tape or you may clear all data from it or erase the program from RAM entirely. The most salient advantage provided by an ESC over specifically vertical applications software is that an ESC provides a reusable framework with which you can compose any specific financial model rather than just be limited to only one statically fixed format for storing, displaying and manipulating numerical data.

\$16.95

\$3.00 SHIPPING AND HANDLING / PROGRAM

A.F.R. SOFTWARE

**- 1605 Pennsylvania Avenue, No. 204
Miami Beach, Florida 33139**

FLORIDIANS ADD SALES TAX

DEALER INQUIRIES WELCOME

ALSO AVAILABLE FOR THE T/S 2068

WHILE THEY LAST...

BRAND NEW **TOP** QUALITY
DOUBLE SIDE DOUBLE DENSITY
DISC DRIVES \$ 99

2/3 HEIGHT 400 KILOBYTE CREAM COLORED

DUAL DRIVE CABINET, TOP QUALITY
HEAVY GAUGE STEEL WITH 5 AMP
SWITCHING POWER CONVERTER **\$ 99**

We want to say *THANK YOU* for the many months of your patience. Our Disc Interface for the TS2068 has 64K of on-board RAM which may be used as a second bank of system memory or a full-blown CPM system. It controls 4 floppy disc drives of any size- from 3 to 8 inches. 1 or 2 sides, single, double, or quad density (8" DD not yet supported). It also includes a complete RGB interface with separated sync so you can *SEE* the High Resolution Color Graphics that this machine produces. It does not include serial or parallel interfaces... you add as many as you like at \$ 69 for 1 Centronics Type Parallel Interface, P/N CP68 \$ 99 for dual independent RS-232 Serial Interface, RS68. The Disc Board is called FD68 and costs \$ 199. Deduct \$ 10 for the first CP68 or RS68. Add \$ 3 / item shipping.

We are now taking orders.



ACME ELECTRIC ROBOT CO.

Box 18093, Austin, TX 78760

(512) 451-5874

2068 CASSETTE CONNECTION

Part II: Loading Tips

In the last issue, I recommended removal of a couple capacitors in the SAVE circuitry of the TS2068. This provides a "brighter" save signal, which makes subsequent loading more reliable. This is all that is required to get many systems working "up to spec," and makes the use of "fast-load" routines possible, even practical. On further research, however, I found that there can still be problems with some systems involving the LOAD portion of the computer-to-tape interface. In this article, I'll discuss the load circuitry, and suggest a few minor changes that you can make to improve reliability if you're still having trouble.

Before I get on with this, I should mention that NOT ALL TS2068's ARE CREATED EQUAL. I had the opportunity to look at a 2068 recently acquired from Games To Learn By (GTLB), and found that the small capacitors directly across the MIC and EAR jacks on the underside of the board were absent on this unit. Also, the two series 1N4148 diodes CR25 and CR26 (discussed later) were bridged with a single identical unit. Furthermore, there are appreciable discrepancies between the Timex schematic and the actual hardware. For example, the SAVE circuit shows yet another capacitor across the MIC output (C72) which is not even marked on the board on either unit I've seen. There are also various other capacitors and wires floating around which look suspiciously like "tack-ons."

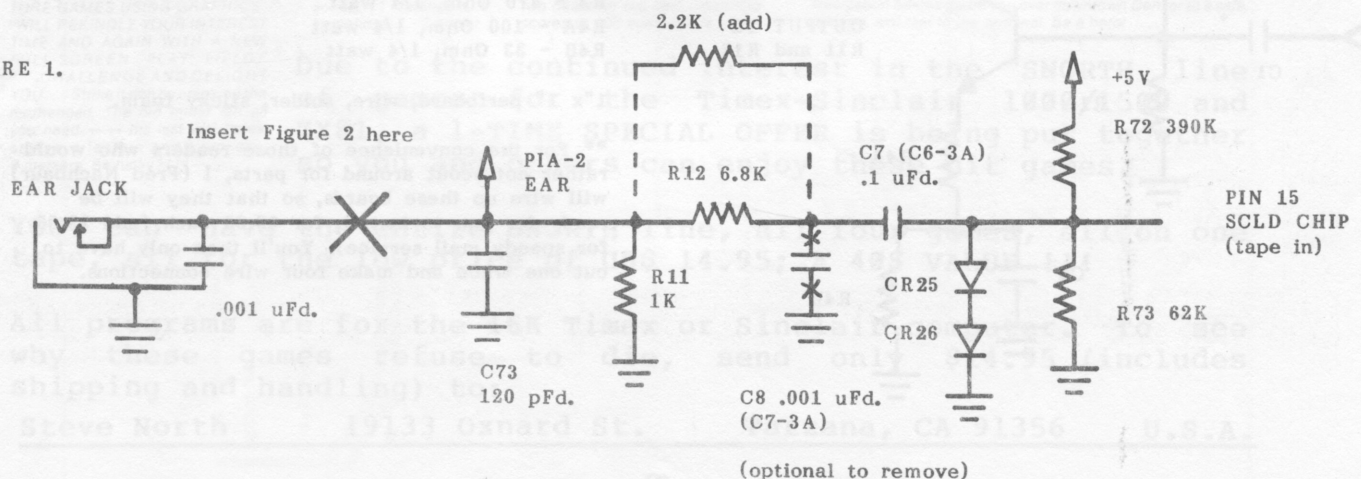
To complicate matters further, it appears that there are at least two discrete board patterns in existence, with different parts callouts for the same components. These alternate parts callouts are shown on the schematic in parentheses with a "-3A" suffix. For example, capacitors C7 and C8 in the LOAD circuitry (discussed later) are C6 and C7, respectively, in the "-3A" version. Both of the boards I've seen have a "-03D" suffix in the number under the "TIMEX 2000" legend and both corresponded with the "-3A" variation on the schematic, so presumably any board designated "-3A" or greater will follow the alternate part numbers on the schematic. The bottom line is that what you read here and elsewhere about the 2068 hardware may or may not jive exactly with your machine.

Now, let's see what we can do to make it easier to "get loaded" on the 2068. The applicable portion of the schematic is reproduced here in Figure 1. As in the case of the MIC jack, there is at least one capacitor (C73) directly across the EAR jack. Again, this (or these) were presumably included to help reduce radio and TV interference. It seems that Timex's philosophy was, "if some capacitors do some good, then more will do more good." Unlike the caps across the MIC port, however, the one or ones on the EAR jack have a negligible effect on the tape signal itself. This is because the impedance looking into the MIC input of the recorder is several magnitudes higher (typically about 50,000 ohms) than the impedance of the EAR output (around 10 ohms). As a result, there is nothing to be gained by removing these capacitors, so you might as well leave it in! Who knows, they might even help reduce interference slightly.

Looking further into the schematic, we find a resistor (R11, 1000 ohms) in parallel with the EAR jack as well. Again, this is so much greater than the output impedance of the tape recorder EAR jack that it has no significant effect on the tape signal. It was probably included to provide the DC return required by some recorders. After that is a little network consisting of two resistors R12 and R73, two caps C8 (C7-3A) and C7 (C6-3A), and two diodes (CR25 and CR26). This is essentially a "diode clamp" circuit, which shifts the signal from, say -3 to +3 volts, to about -4 to +1.2 volts. Also, it provides a low-frequency (LF) corner of about 230 Hz., and a high frequency (HF) corner near 23 kHz.

My reason for digging into this was that I sometimes couldn't get some tapes to load, even at maximum volume setting on my 6 volt TS2020 or Minisette IX's. A quick measurement showed that my line voltage was somewhat low, around 110 volts. My present home-sweet-home has long secondary power lines and electric heat; I found that some tapes would load ok with the heaters unplugged, but wouldn't "take" if the AC line voltage dropped below the rated 120 VAC. A colleague reported similar problems, especially when trying to use a fast-load program to speed up the tape processes. If you're having similar border-line sensitivity problems with tape loading, the following modification may quite possibly handle it.

FIGURE 1.



(optional to remove)

2068 LOAD Circuit Mods

- 1: Open the 2068 by removing the seven screws and disconnect the keyboard tail. (Thankfully, the KB tail on the 2068 is considerably more substantial than the ones on the ZX81/TS1000!)
- 2: Locate the 6800 ohm resistor, R12 (blue-grey-red-gold). This is about 1 cm. below and slightly to the left of the C10 capacitor you removed last issue. Bridge this resistor with a 2200 ohm, 1/4 watt unit (red-red-red-gold). This will make the mod easy to remove if it doesn't do the trick for you. Alternately, you may clip the 6800 ohm resistor and replace it with an 1800 ohm unit. This increases sensitivity, which in my case was enough to allow loading tapes that wouldn't quite "take" even at full blast. It also raises the LF corner, making the system yet more immune to low-frequency garbage.
- 3: I also removed capacitor C8 (C7-3A), even though doing so is not strictly necessary; the HF corner created by this cap is high enough to avoid trouble with the standard tape routines. However, it may conceivably get in the way if you later use VOTEM or other V-F (voltage to frequency) analog interfaces, or if you decide to experiment with fast-load routines. Although the 2068 (about 1500 baud) is a lot faster than the ZX81 (at about 300 baud), a good tape recorder will support even faster loading; up to about 3500 baud, or 440 bytes per second.
- 4: Reconnect the KB tail, and set the board back into the case. Test your modification. If you're satisfied that you now have enough sensitivity, screw the two case halves back together.

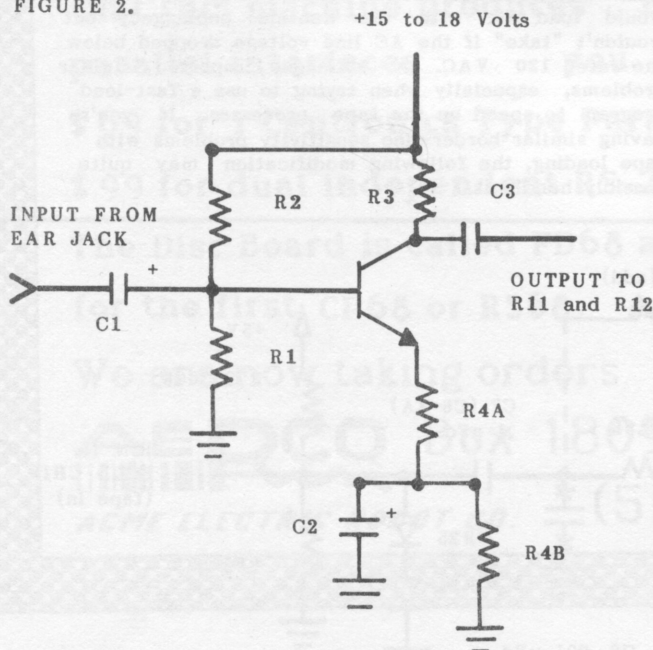
This will handle many loading problems. Aside from the comparatively low sensitivity of the 2068's load circuitry, there's not much for which it can be faulted. Unlike the ZX81, the SCLD custom chip apparently contains a comparator to square up the

signal. As a result, adding a comparator or Schmidt-trigger conditioner probably won't have a noticeable effect. However, if you have a tape recorder with a 6 volt supply, chances are you might still have trouble with sensitivity. If so, here are a few options:

- 1: You can use VOTEM to help out, but you'll need to make a couple modifications. First, you'll have to supply a 9-volt power source such as a "transistor radio" type battery. Then, you'll have to connect the junction of R1 and R6 to +9V instead of +5, to give you a larger output swing. One caution, however; this modification may make it unusable with the ZX81/TS1000, as these prefer a lower tape signal voltage and some of them balk if it's too high.
- 2: Get a recorder with a higher supply voltage, e.g. 7.5 or 9 volt. This will allow a higher output swing.
- 3: Replace your present 6V AC adaptor with a 7.5 volt unit. These are available from Radio Shack and other sources. Try to get it on the condition that you can return it if it doesn't do the job. The slightly increased supply voltage will not harm well-designed recorders, but may be just enough to push you "over the hump." This option, if it works, is a lot cheaper than buying a new deck.

You might experiment with the number of diodes in the clamp. (Be warned! You are dealing directly with the SCLD chip. DON'T BLOW IT!-ed.) This affects the positive clamp level, at the rate of about .6 volts per diode. As mentioned above, the units from GTLB have the two diodes (CR25 and CR26) bridged with a single one. I tried this on mine, and found that it actually seemed to make matters worse. I also tried adding an extra diode in series, and found no noticeable change. You might also try increasing C8 (C7-3A) to, say 0.22 uF., especially if you reduced R12.

FIGURE 2.



PARTS LIST - 2068 LOAD AMP

- C1-C3 - 1uFd., 16V tantalum
- C2 - 47 uFd., 6V electrolytic
- Q1 - general purpose audio transistor
2N3904, 2N4401, etc.
- R1 - 910 Ohm 1/4 watt
- R2 - 4700 Ohm, 1/4 watt
- R3 - 470 Ohm, 1/4 watt
- R4A - 100 Ohm, 1/4 watt
- R4B - 33 Ohm, 1/4 watt

1"x 1" perfboard, wire, solder, sticky foam.

** For the convenience of those readers who would rather not scout around for parts, I (Fred Nachbaur) will wire up these boards, so that they will be ready for you to install, for \$6.00 each (add \$3.00 for speedy mail service). You'll then only have to cut one trace and make four wire connections.

The Final Solution

If you reduced the value of R12 as described above and still don't get enough sensitivity with your deck, and if the other "fixes" don't appeal to you, your best bet is to add a linear amplifier to boost the signal to the computer. Not much gain is needed, so a single-transistor "class A" stage is all you need. The circuit of Figure 2 is the answer that my ZX81 "CE AMP" program came up with. It has a voltage gain of about 3, and a P-P output swing of about 10 volts. Cost in parts is around \$4, depending on how much of the circuit you have in your junk box. On my machine, it allows most tapes to load reliably with a volume setting anywhere between 4 and 10.

Wire it up on a 1"x1" piece of perf board, cut the trace to the EAR jack (underside of board) and connect the EAR jack to the input. Connect the output to the left side of R12. Get the +15 volts power for the circuit from the on-off switch (SW2), the leg closest to the front edge of the board. A good place to get the ground is from the grounding strap soldered to the top of the video-section enclosure. (If you want to get fancy, you can make use of the expansion port area. A1-top left is ground, A2 is the EAR jack and B3 is the +15V power.-ed.) The transistor can be virtually any NPN silicon type capable of at least 200 mW. dissipation and having a current gain (beta) of 50 or higher; e.g. 2N3904, 2N4401, etc. (Beware of the Radio Shack "2N3904 grab-bag." These are, in my experience, pretty lousy.)

Mount the board on top of the keyboard diodes (in front of the keyboard connector) using sticky-foam; watch for shorts. If you wish to use one of the little "Walkman"-type recorders with their 3V

supplies, increase the gain to about 4 by making R1 = 1000 ohms, R4A = 75 ohms, R4B = 62 ohms. C1 and C3 should be 16V tantalum units, C2 may be an aluminum electrolytic rated 3V or up.

Load Amp for TS1000

You can use the one-transistor load amplifier circuit with the ZX81/TS1000/TS1500 by making the following changes in component values: *

R1 = 1000 ohms	R4B - not used	* As for the 2068,
R2 = 4700 ohms	C1 = 1 uF.	pre-wired boards
R3 = 160 ohms	C2 - not used	are available from
R4A = 33 ohms	C3 = 3.3 uF.	the author for \$6

These changes are necessary because of the different supply voltage (9V instead of 15V) and input resistor value (220 ohms instead of 1000 ohms) as compared to the TS2068. Gain is set at about 3, and P-P voltage swing is just under 5 volts.

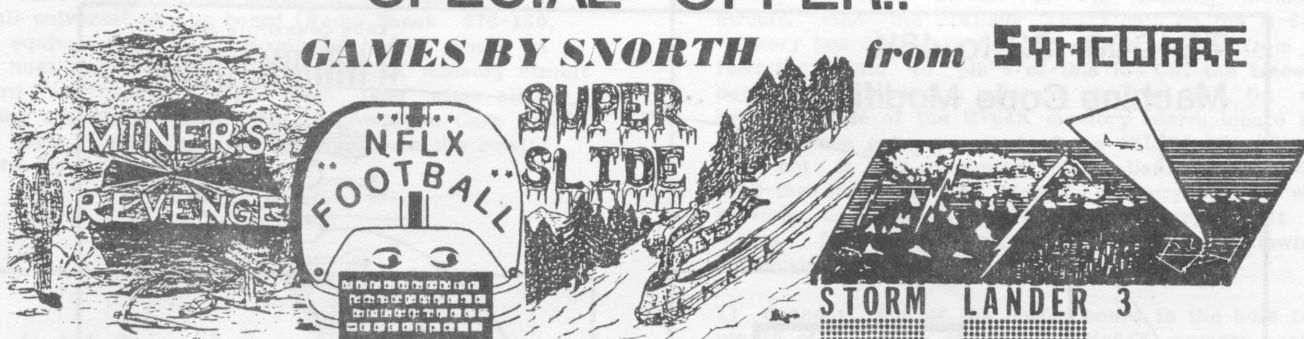
If you're using fast-load routines such as Z-XLR8, SDS, or Q-SAVE, it is recommended that you change capacitor C11 (located near the MIC jack on the ZX81 board) from 47 nF. to 20-22 nF (.02-.022 uF.). This increases the HF corner of the SAVE circuitry from about 3.5 kHz. to over 7 kHz. and provides a brighter signal. I did this simply by breaking the existing capacitor in half horizontally, using a pair of wire nips; works fine, but not a "guaranteed" procedure. After doing this mod, I found that I could run SDS (similar to Z-XLR8 at top speed) even without a pre-conditioner.

So there you have it. That should wrap it up, at least from a hardware standpoint. In a future issue we'll try to get you a flexible, variable fast-load routine. 'Till then, "good loads, fair weather!"

SPECIAL OFFER!!

GAMES BY SNORTH

from SYNCWARE



A CLASSIC ARCADE/ADVENTURE HIT YOU'LL AGREE WITH THE REVIEWS-"EXCELLENT EXAMPLE OF THE NEW GENRE OF ADVENTURE GAMES USING GRAPHICS" "WILL REKINDLE YOUR INTEREST TIME AND AGAIN WITH A NEW FULL-SCREEN PLAY FIELD." "CHALLENGE AND DELIGHT YOU..." Strike it rich by cracking the motherload. The old miner left all you need — his last will, flares, spikes, a lantern and a pickaxe. 9 markers. Be lucky!!!!

HIT GAME!!! THE FIRST FULL ACTION GRAPHICS FOOTBALL GAME. Not a text game. Fast M.C. action. 11 defenders. 8 formations. 1 - 2 player.

WIN A GOLD MEDAL. SUPER FAST M.C. ACTION. INCREDIBLE 8 MILE BOBSLED COURSE. Survive the qualifying run, then race two heats for the best combined scores vs 100 competitors.

MULTI-SCREEN EXCITEMENT! MAKE AN EMERGENCY LANDING AT DENVER'S STAPLETON FIELD IN YOUR COMMERCIAL JET. 5 screens of graphics show your progress through storm clouds, other aircraft, turbulence, mountains. Navigation beams guide you over downtown Denver to a safe landing, and taxi to the terminal. Be a hero!

Due to the continued interest in the SNORTH line of games for the Timex-Sinclair 1000/1500 and ZX81, a 1-TIME SPECIAL OFFER is being put together so you and others can enjoy these hit games.

YOU can have the entire SNORTH line, all four games, all on one tape, all for the low price of US\$ 14.95; A 40\$ VALUE !!!

All programs are for the 16K Timex or Sinclair computer. To see why these games refuse to die, send only \$14.95 (includes shipping and handling) to:

Steve North 19133 Oxnard St. Tarzana, CA 91356 U.S.A.

HARDWARE PROJECTS

Run TS1000 Machine Code in High Memory

John Oliger

ZX81ers don't despair! This next trick was originally published in Syntax Quarterly, Summer 1983. It will allow you to use the 32-48K RAM area for machine code, if you have the Memotech or JLO 64K boards (and maybe some others, but not the Byte-Back UM).

It is necessary to add a little extra circuitry to separate out the video system from the dynamic RAM system, as Sinclair never dreamed of a ZX81 with more than 16K. Very simply, you cut the M1 not trace between the Z80 (pin 27) and the ULA (pin 10). Connect pin 2 to the Z80 side and the output on pin 8 to the ULA side. Again, use an ohm meter to chase down the A14, A15, Vcc (5 volts) and ground lines. They can all be picked up at or near the edge connector (except pin 8), but you may find some more convenient places. The diagrams show some very good places to pick up these traces. Be sure to remove all ICs (if possible) before doing any soldering.

Any relocatable programs will run here and not be affected by NEW or reset. ZXLR-8, Delphic toolkit, compiled programs with the Bob Berch Compiler, Hot Z and an assortment of other utilities will work up there and NOT interfere with any 16K program on the market.

Additional Byte-Back Memory Pack Notes:

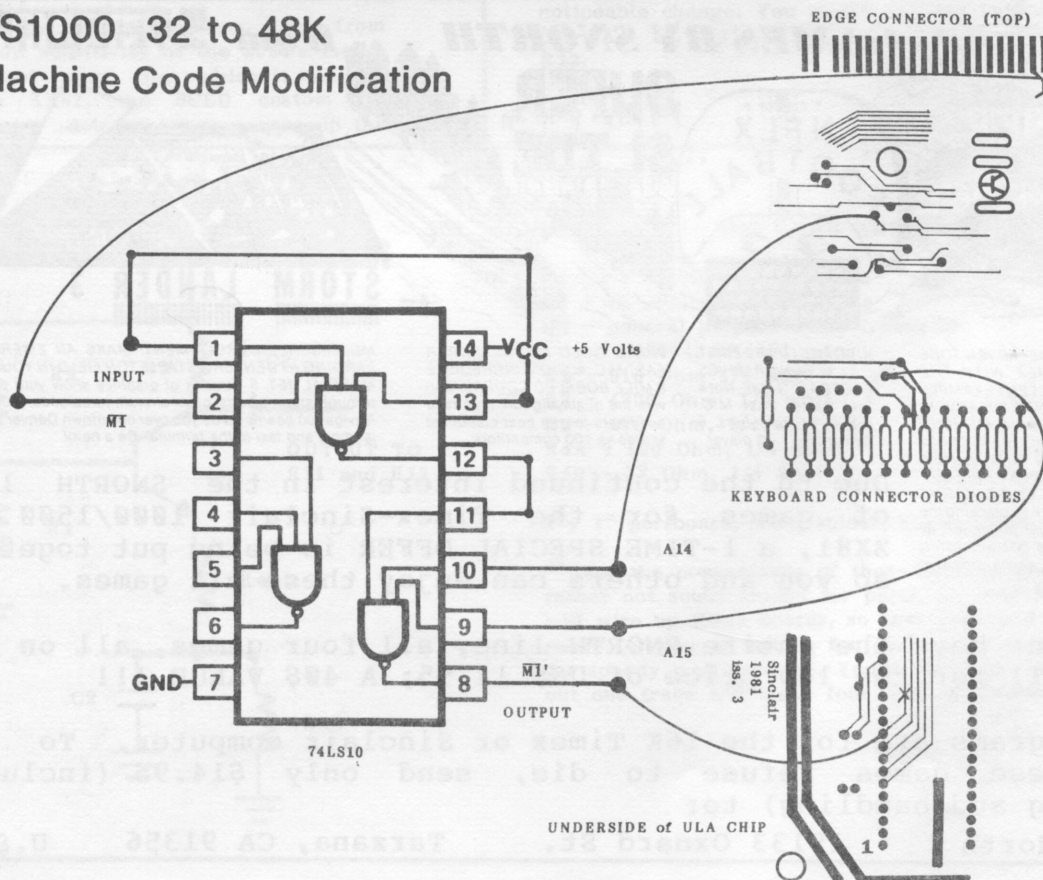
Jeffrey D. Moore
602 S. Mill Street
Louisville, Ohio 44641

Thanks to John Oliger's NOT M1 decoding circuit, many new and excellent pieces of software are coming into the market place that make use of machine language routines in the 32K-48K (8000-BFFF hex) region of memory. Some examples of these would include Hot Z-II by Ray Kingsley and Memotext in RAM, Version 3, modified and marketed by Fred Nachbaur.

However, there is one catch! It is best summed up by Ray Kingsley in his Hot Z-II User's Notes. "If you have a suitable memory...Oliger or Memotech or possibly another: Not a Byte-Back M-64 (or UM-64K memory-JM)...and you make the Oliger modification to your computer, you can run machine code in the 32K-48K block..."

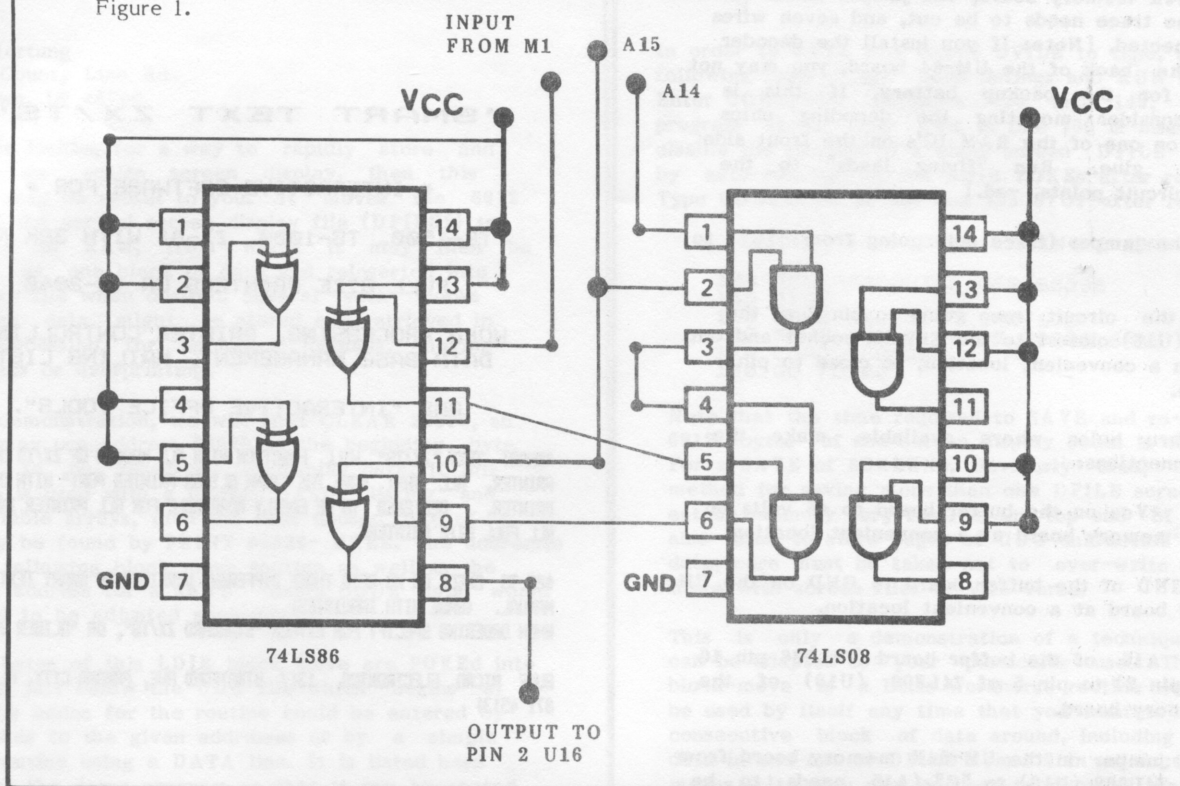
I contacted Mr. Oliger, and he sadly confirmed that it was true, a Byte-Back M-64 or a UM-64K memory would not work with the NOT M1 decoding circuit. However, our conversation, and subsequent correspondence, led Mr. Oliger to develop the two chip

TS1000 32 to 48K Machine Code Modification



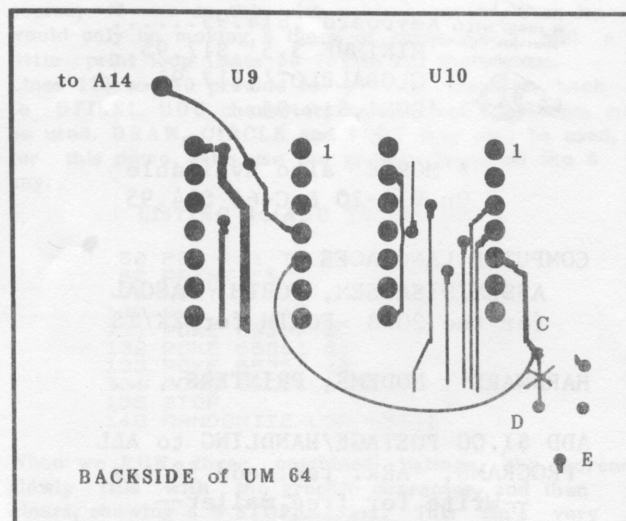
MODIFICATION for the BYTE BACK UM-64 RAM PACK

Figure 1.



buffer circuit shown in figure 1. I implemented and tested it in the Byte-Back M-64 Memory Pack, and found that it works great!

The buffer circuit itself should be constructed on a small universal circuit board (Radio Shack 276-150, or equiv., works well). Mount the buffer circuit on the heat sink side of the M-64/UM-64K memory circuit board with double-faced foam tape after all the wiring and inter-connections are complete. Care MUST be taken to fully electrically insulate the small board from the main board.



M-64 Hook-up

Modification of the M-64K memory board requires that two circuit traces be cut and seven wires installed to add the new circuit to the existing memory circuit. Find the 74LS08 (U12) chip on the M-64K memory board. Locate the circuit trace going from a feed-thru hole to pin 6 of this IC. Cut the trace between the feed-thru hole and pin 6 of U12. On the heat sink side of the M-64K memory board, locate the circuit trace going to pin 4 of the 74LS32 IC (U11) and cut it at a convenient place. Using the plated feed-thru holes on the M-64K memory board where possible as solder points (not necessary, but it makes for a neater job), make the following connections:

- 1) Connect Vcc of the buffer board to the hole for pin 16 of the spare chip on the M-64K memory board.
- 2) Connect GND (ground) of the buffer board to the hole for pin 8 of the spare chip on the M-64K memory board.
- 3) Connect A15 of the buffer board (74LS08 pin 2 and 74LS86 pin 10) to pin 5 of the 74LS08 (U12) on the M-64K memory board.
- 4) Install a jumper on the M-64K memory board from pin 5 of the 74LS08 (U12) to pin 4 of the 74LS32 (U11).
- 5) Connect A14 of the buffer board (74LS08 pin 1) to pin 5 of the 74LS32 (U11) on the M-64K memory board.
- 6) Connect NOT M1 of the buffer board (74LS86 pin 12) to pin 4 of the 74LS08 (U12) on the M-64K memory board.
- 7) Finally, connect the output of the buffer board (74LS86 pin 8) to pin 2 of the 74LS244 IC chip (U16) closest to the 74LS08 chip (U12) on the M-64K memory board.

UM-64 Connections

On the UM-64K memory board, one jumper needs to be removed, one trace needs to be cut, and seven wires must be connected. [Note: If you install the decoder board on the back of the UM-64 board, you may not have room for the backup battery. If this is important, consider mounting the decoding chips upside-down on one of the RAM IC's on the front side with crazy glue. Run "flying leads" to the appropriate circuit points. -ed.]

- 1) Remove the jumper (trace cut) going from "C" to "D".
- 2) Locate the circuit trace going to pin 2 of the 74LS244 IC (U16) closest to the EPROM socket and cut the trace in a convenient location, as close to pin 2 as possible.

Using feed-thru holes where available, make the following connections:

- 3) Connect +Vcc on the buffer board to +5 volts on the UM-64K memory board at a convenient location.
- 4) Connect GND of the buffer board to GND on the UM-64K memory board at a convenient location.
- 5) Connect A15 of the buffer board (74LS86 pin 10 and 74LS08 pin 2) to pin 5 of 74LS08 (U10) of the UM-64K memory board.
- 6) Install a jumper on the UM-64K memory board from pin 5 of the 74LS08 (U10) to "C" (A15 needs to be connected to pin 5 of U9, through "C" is the easiest way to get there).
- 7) Connect A14 of the buffer board (74LS08 pin 1) to pin 4 of the 74LS32 (U9) of the UM-64K memory board.
- 8) Connect NOT M1 of the buffer board (74LS86 pin 12) to pin 4 of the 74LS08 (U10) of the UM-64K memory board.
- 9) Finally, connect the output of the buffer board (74LS86 pin 8) to pin 2 of the 74LS244 chip (U16) closest to the PROM socket on the UM-64K memory board.

Be sure to double check all of your connections to insure that they are correct and that you have not created any solder bridges. Make sure the buffer board is secure to the M-64K memory board and that they are insulated from each other.

If you haven't already done so, make the modifications to your computer for NOT M1 Decoding per John Olliger's article and test that circuit. When all is well, power down your computer and connect the modified M-64K/UM-64K Memory Pack to it. Power back up. If that old familiar little "K" cursor shows up on the screen, you're over half way home. If not, immediately power down and re-check the buffer circuit and its connections for shorts and improper wiring. When everything checks out and powers up O.K. with the modified memory pack connected, enter the following commands: POKE 40000,201 followed by PRINT USR 40000. The screen should return 40000. This is a fair indication that everything is pretty much as it should be, but it is not foolproof. The "acid" test is to now load or write a program that runs MC in the 32K-48K region of memory and try it. It should now run correctly without any problems.

"SMART TEXT ZX/TS"

* INTERACTIVE SOFTWARE FOR *
TS-1500, TS-1000, ZX-81 WITH 32K RAM
FULL SIZE PRINTERS OR TS-2040
WORD PROCESSING, PRINTER CONTROLLING,
DATA BASE MANAGEMENT, MAILING LIST,
AND "INTERACTIVE OFFICE TOOLS".

"SMART TEXT ZX/TS" WILL FUNCTION WITH ALL MODELS OF ZX/TS USING THE 2040 PRINTER, ALL THAT HAVE THE "JOHN OLLIGER PRINTER PORT" WITH OKIDATA ML-82A PRINTER. DESIGNED TO BE EASILY ADAPTABLE FOR ALL PRINTER INTERFACES, AND ALL FULL SIZE PRINTERS.

\$29.95 CHECK OR MD GETS THREE DIFFERENT VERSIONS OF SMART TEXT, AND 35 PAGE MANUAL. (BASE WITH INQUIRIES)
WHEN ORDERING SPECIFY FOR EITHER "STANDARD ZX/TS", OR "OLLIGER VDP".

GULF MICRO ELECTRONICS, 1317 STRATFORD AVE, PANAMA CITY, FL 32404. (904 871 4513)

WE ARE YOUR TIMEX/SINCLAIR SOURCE

In our "HAM-HACKER"(tm) SERIES :

For the 16K ZX/TS.....
'MORSE/2K', \$14.95.....
'MINIMUF 3.5', \$17.95..
'SUN', \$14.95.....'BEAM
HEADING', \$14.95...'C E
AMP', \$19.95



For the 2068...'MORSE/
Keyboard', \$14.95.....
'MINIMUF 3.5', \$17.95..
'GLOBALPLOT', \$17.95...
'SUN', \$14.95

*'MORSE' also available
On VIC-20 & C-64 \$14.95

COMPUTER LANGUAGES :

ASSEM/DISASSEM, FORTH, PASCAL
for the 2068 -FORTH for ZX/TS

HARDWARE : MODEMS, PRINTERS.....

ADD \$1.00 POSTAGE/HANDLING to ALL
PROGRAMS, Ark. res. add 5% tax
--Write for free mailer--

HAWG WILD Software™
P.O. Box 7680 • Little Rock, Arkansas 72217

QUICK SCREEN DISPLAY 2068

Robert Hartung
2416 N. County Line Rd.
Huntertown, IN 46748

If you are looking for a way to rapidly store and retrieve an entire screen display, then this approach may be useful to you. It moves the 6912 bytes of the normal screen display file (DFILE1) to an address in RAM, from which it may then be retrieved as one block of data and reinserted into the display file when desired. Several such blocks of display data might be stored and retrieved in rapid succession for animated displays, moving backgrounds or overprinting.

For a demonstration, we will first CLEAR 29999, so that we may use address 30000 as the beginning byte of RAM to be reserved for display data. If this technique is used with a longer BASIC listing and large variable arrays, then the first unused byte of RAM may be found by PRINT 65536- FREE. The addresses in the following block move routine as well as the beginning address for a SAVE "name"CODE m,n would then need to be adjusted accordingly.

The 12 bytes of this LDIR block move are POKED into a location just below the UDG file which begins at 65368. The codes for the routine could be entered by direct pokes to the given addresses or by a simple loader routine using a DATA line. It is listed here as part of the demo program so that it can be noted as to what each step does and where changes can be made to adapt it to other values.

LISTING 1

```
80 POKE 65355,1
81 POKE 65357,48
82 POKE 65358,24
83 POKE 65359,17
84 POKE 65360,48
85 POKE 65361,117
86 POKE 65362,33
87 POKE 65363,0
88 POKE 65364,64
89 POKE 65365,237
90 POKE 65366,176
91 POKE 65367,201
100 RANDOMIZE USR 65356
```

When this listing is RUN as given, the 6912 bytes of DFILE1 (display and attributes) would be transferred to th storage address beginning at 30000d. Of course, if we do this with a blank screen, then we would only be moving a block of zeros. Let's add a little print loop (lines 50-70) to fill the screen. Lines 130 to 140 provide for a block transfer back to DFILE1. UDG characters and normal characters may be used. DRAW, CIRCLE and PLOT may also be used, but for this demo, let's use the graphic found on the 6 key.

LISTING 2--ADD TO LISTING 1

```
50 FOR i=1 TO 704
60 PRINT " ";
70 NEXT i
130 CLS
131 POKE 65360,0
132 POKE 65361,64
133 POKE 65363,48
134 POKE 65364,117
135 STOP
140 RANDOMIZE USR 65356
```

When we RUN these combined listings, the screen slowly fills with the graphic character, and then clears, showing a 9 STOP, 135:1. This isn't very impressive, yet. Now type CONTINUE and ENTER. Is that fast enough?

In order to SAVE this stored DFILE to tape, add the following lines to the above listings and RUN again. Enter CONTINUE at lines 135 and 145, when the program stops. The REM in line 100 is inserted to disable the storage of a blank screen DFILE caused by auto-running of the LDIR POKES after reloading. Type CONTINUE at the line 135 STOP after reloading.

LISTING 3--ADD TO LISTING 1 AND 2

```
100 REM RANDOMIZE USR 65356
145 STOP
150 SAVE "quick" LINE 170
160 SAVE "quick"CODE 30000,6912
170 LOAD "CODE
180 GO TO 80
```

Note that the time required to SAVE and re-LOAD the 6912 bytes of the screen display codes is the same for a SAVE of SCREEN\$. Obviously then, using this method for saving more than one DFILE screenful will eat up memory very fast. If the top end of RAM is also used to store pages of UDG characters or other data, care must be taken not to over-write any of these with screen files or vice versa.

This is only a demonstration of a technique which can be adapted to your particular use. The LDIR block-move is a little workhorse routine which may be used by itself any time that you want to move a consecutive block of data around, including moving data in and out of DFILE 1 and 2 in enhanced display modes. It may also be used to move all or selected blocks of ROM codes up into RAM where you can study the effects of making various changes. It should be noted however, that some of these rom routines will still call to their original addresses, and non relative jump instructions will need revision for their new locations as well.

HEXCODE	LABEL	MNEMONIC
013018	SAVE	LD BC,1830
113075		LD DE,7530
210040		LD HL,4000
EDB0		LDIR
C9		RET
013018	SHOW	LD BC,1830
110040		LD DE,4000
213075		LD HL,7530
EDB0		LDIR
C9		RET

TEACHERS * TEACHERS * TEACHERS * TEACHERS

Finally a truly useful gradebook! T/S GRADER handles:

1. Multiple classes
2. Machine Code speed
3. All data on ONE screen
4. Weighted/Unweighted avgs
5. Bonus/demerit points
6. Alphabetizing
7. Timex printer output
- And a whole lot more!

VERY HIGH CAPACITY

16K TS1000 can track from 279 pupils with 5 grades each up to 64 pupils with 66 grades each

Easily converted to 64K with 7 times more capacity!

TS2068 has 5 times the capacity of 16K version

TS GRADER 1000=\$15.95 or TS GRADER 2068=\$19.95

\$1 (refundable on order) for info sheet

Robert Fischer; 221 Scoggins St; Summerville, Ga 30747

EPROM PROGRAMMER

PART II Fixing Your Home ROM

John Oliger
11601 Whidbey Drive
Cumberland, IN 46229

The programs of listings 1 and 2 in Part 1 got you programming 27128's on the standard TS2068 computer with the supplied home ROM (U16). There IS, however, another way to get around the problem which is much better in the long run, if you plan on programming very many 27128 EPROMs. And what is that? Why, correct the home ROM by transferring it into RAM, changing the code, and then programming a 27128 to replace it! How do you change it? Use Hot Z 2068. What are the required changes? They are listed below:

```
002B 84    DATA    ;This is a new little
002C 87    DATA    table added.
002D 8B    DATA
002E 8D    DATA
002F 92    DATA

006D 2801  JR Z,0070 ;This is a correction to an
                        NMI bug

37B8 D9    EXX      ;Get exchange registers
37B9 212B00 LD HL,002B ;Form pointer into
37BC 85    ADD A,L   ;the new table in the
37BD 6F    LD EL,A   ;HL register pair
37BE 6E    LD L,(HL) ;Get low byte of desired
                        constant address
37BF 2636  LD H,36   ;High byte of constant
                        address is 36h
37C1 D9    EXX      ;Return it in HL'
37C2 AF    XOR A     ;Let A=0 & clear carry flag
37C3 C9    RET       ;Done, so return

37C4 00    NOP       ;Extra byte
```

The above subroutine does the same thing that the routine it replaces did, but it does it without writing garbage to itself, and even does it more efficiently. If you run the benchmark speed tests given in recent issues of Creative Computing both before and after these changes, you will find that the 2068 gains 5 seconds with these changes. Not enough to lie awake at night about, but faster nonetheless. Also, the changes listed above correct an old error in the NMI handler in the Sinclair Spectrum. [The NMI line is used by peripherals like disc drives, etc., in order to run efficiently. - ed.] With this change, if you know any m/c, you can examine the NMI routine and, I am sure, find out how to use it. (How would you like to add a break key that could stop ANYTHING, including non-breakable programs and run-away machine code!) Now, getting back to the changes necessary to make these corrections to the home ROM. Can you simply mount the new 27128 EPROM in the home ROM's socket without any hardware changes? No, I'm afraid not. Can you simply make the trace cuts and wire jumps listed in the TS2068 Technical Manual? You can, BUT if you do your 2068 EPROM programmer will no longer work

correctly, because this will take away the RD NOT decoding required for the programmer. So, here's what you do to mount this EPROM in the U16 socket and STILL be able to use the 2068 Programmer. You simply use that extra gate we left open for this purpose in a previous episode.

Remove all screws from the bottom of the computer case and lift the case top from the computer, carefully unplugging the keyboard cable while doing so. Find jumper resistors W1 and W2 near the center of the board and clip both of these from the board.

Now, using wire wrap wire and small gauge solder, make the following connections to and from this chip. Any pin number not listed is left unconnected.

Pin 1 to left W1 pad.

Pin 2 to left W2 pad.

(as per instructions in SWN 2:3, pg. 19)

Pin 3 to right W1 pad.

Pin 14 to right W2 pad (AND Vcc +5V as in SWN 2:3; a good place to get this is pin 9 of U13.)

Pin 7 to U12 pin 18. (ground)

When soldering these wire-wrap wires to the wide part of the LS/HC32 chip, solder quickly to avoid damage to the chip. The EPROM mod is now done. You can now install your 27128 in the U16 socket. (And you didn't even have to make a trace cut!)

If you desire to install a 2764 in the EXROM (U20) socket, you CAN do it the way Timex says in the Tech manual without any conflicts. But a better way is to use the same socket as was used for Spectrum compatibility in Issue 2:4 of SyncWare News. Cut off pin 1 and pin 27 of the EPROM socket, and solder small jumper wires from pin 28 to the bases of each pin (1 and 27). This way, you again, don't have to make any trace cuts!

Plug your keyboard cable back in and reassemble your case and you're ready to go. With these changes made, and the EPROM safely inside your TS2068, you no longer have to use those special programs when programming 27128 EPROMs. You can now use a simple FOR/NEXT loop as shown for the 2764, with the base programmer address changed and the capability of 16K storage instead of just 8K.



GET BACK TO BASICS with BASICALLY PROGRAMMING

asically
rogramming

2528 W. Olive
Fullerton, CA 92633
(714) 738-0666



TS2068 Programs
MULTI-DRAW 2068.....\$24.95
Softsync Assembler.....\$19.95
Voice Chess.....\$24.95
Personal Accountant.....\$24.95

TS1000 Programs
EZ-HEX.....\$12.95
Programmers Toolkit.....\$14.95
ZX Pro-File (16K to 64K).....\$16.95
ZX Data Finder.....\$14.95
Z-Wryter (16K to 64K).....\$12.95
Profit Plan.....\$12.95
Quest for the Holy Grail
& the Elusive Mr. Big.....\$17.95
JD Black Star.....\$ 9.95
Torpedo Attack.....\$ 9.95
Games Pack (2K).....\$14.95

Please include check or money order. Calif. residents add 6% sales tax. Please add \$2.50 shipping and handling. No COD's - US funds only.

SEND NOW FOR FREE CATALOG!

BP - We have not abandoned basic computer users!

Verify Your EPROMS

Because the address space used by this programmer is shared with the 2068 home ROM, and to keep both the hardware and software involved as simple as possible, this programmer does NOT include verify circuitry. To verify an EPROM you must either run a checksum on the data to be stored and run the same checksum routine on the EPROM after programming, (Hot Z 2068's VERIFY function is good for this), or store the data on tape to be loaded in later for a direct comparison with the programmed EPROM in a User Cartridge board. An example of this follows:

You have programmed a 27128 EPROM with data that was stored in RAM from 49152 to 65535 (top of memory). You have saved this data to tape with a <SAVE "data" CODE 49152, 16384> statement. Install the programmed EPROM in the cartridge board, mapped from 32-48K via diodes to the decoder's "32" and "40" outputs. Clear out the top 32K of memory with a <CLEAR 32767> command, then load the data back into the top of memory with <LOAD "" CODE>. Now key in the following comparison/verification program:

```
10 OUT 244,48
20 LET x=49152:FOR n=32768 TO 49151
30 IF PEEK n<> PEEK x THEN PRINT n;"=";
   PEEK n,x;"="; PEEK x
40 LET x=x+1: NEXT n
```

If the little program finishes with a clear screen, then the EPROM is verified as being 100% exactly like the data. If there is something on the screen, then you will see what is, and what should have been on the EPROM, and where.

There is really not much need to verify a BASIC program stored on EPROM. If the program RUNs correctly without errors, then you can be certain it is ok. You will find that the only times an EPROM will not verify correctly are when:

- 1) There was an error made in keying in the burner program itself. (Usually resulting in EVERYTHING being wrong.)
- 2) The EPROM is defective. (Lots of times a single bit of all locations will not program.)
- 3) The EPROM was not completely erased.

You can verify that an EPROM has been erased with the following routine (for 27128):

```
10 OUT 244,48
20 FOR n=32768 TO 49151
30 IF PEEK n<>255 THEN PRINT n
40 NEXT n
```

It is not a bad idea to do this to all erased EPROMs.

Programmer Theory

Note: It is certainly not necessary to understand the following details on how this circuit works to build and use this programmer. If, when reading the text below, you find yourself "lost," there is no need at all to be concerned. But, if you find yourself understanding part or all of it, then so much the better! Generally speaking, the more you understand of the workings of a piece of hardware

the more likely you are to be able to use it to its fullest potential.

NOR gate U1 and miniature switch SW1 form an address decoder for the address range from 0 to 16383 (with SW1 in its "128" position) or from 8192 to 16383 (when SW1 is in its "64" position). If MREQ NOT, WR NOT, A15, and A14 are all logic low, and (with SW1 in the "128" position) A13 is a logic high, an active high signal is generated at U1 pin 5. With SW1 in its "128" position, A13 is ignored, making this high pulse appear regardless of A13's state. (I.e., we don't care, in the "128" position, if the memory write is to the 0-8191 chunk or the 8192-16383 chunk.)

This active high pulse is applied to both U2 pin 16, and U3, U4 and U5 pin 11. [Note: The first installment had a typo for this, in the section "The Smoke Test," line 9. This should read, "... while monitoring pin 11 of U3....", not pin 1. ed.] The rising edge of this pulse on U3-U5's pins 11 causes these flip-flops to transfer and hold the current data and address state of their inputs to their outputs. These outputs are applied to the EPROM being programmed, and will remain in this "frozen" state until another write (POKE) to this decoded address space is performed.

Meanwhile, that active high pulse from U1 pin 5 has also been applied to the 555 timer U2 at pin 6. This timer (if you are familiar with the typical wiring of the 555 as a one-shot), is wired somewhat unconventionally so as to respond to an active high trigger pulse and output an active low timed pulse. D1 was added as part of this unconventional wiring, and the timing components (C1 and R1) have been adjusted in value for the correct 50 ms. pulse. Note that the normally-wired one-shot's equation for this IC ($T=1.1 \cdot R \cdot C$) is no longer accurate.

After the triggering pulse is applied to U2's input at pin 6, there is a typical delay of 100 ns before the IC's output at pin 3 goes active low. This delay easily satisfies the EPROM's stable address/data line requirements before allowing its PGM NOT input to go active. This timed, 50 ms pulse from U2 pin 3 is applied to the EPROM's PGM NOT pin 27.

Now the EPROM will take the state of the latched data inputs on its pins 11-19 and store it permanently at the location within itself, pointed to by its latched address inputs if:

- 1: The EPROM's Vpp has 21 VDC applied to it via an external power supply such as the Oliger Vpp Supply set at "Vpp" and "21."
- 2: The data and address lines are stable for the duration of the 50 ms pulse, plus a little more time for the chip to respond to the pulse going inactive. This is accomplished via the latches and the PAUSE 3 BASIC statement in the burner program which keeps the loop from POKEing this address space for 50 ms plus the time used by BASIC in executing the program itself.

The programmer is then ready to accept another byte, and the process continues until the EPROM has been fully burned.

BASIL'S COMPENDIUM

Hexadecimal & 256-imal

Basil Wentworth
1413 Elliston Drive
Bloomington, IN 47401

This chapter will introduce the concept of hexadecimal notation, and what I call "256-imal". But first, the "fun" program.

This program will let you find the memory location of the first byte of any line. Just set the cursor at the line you're interested in, tell the computer to PRINT (notice: not RAND) USR 16514, and voila! There's the memory location of the line number. The first digit of the line number, that is--as you'll learn later on, the computer stores the line number in two bytes.

If you want, you can use this program instead of PROGTOP (see chapter 1). Just give the command POKE USR 16514, 118.

You'll notice that the instructions for building the "fun" program are getting shorter and shorter. As you get more experience, I'm counting on you to fill in the missing details by yourself. So this time you get only a repeat of the loader program (Figure 5-1) and the final listing of the machine code routine (Figure 5-2).

FIG 5-1. THE LOADER PROGRAM

```
5000 LET F=16513
5010 LET F=F+1
5020 PRINT F;" "
5030 IF PEEK F<>27 THEN GOTO 506
5040 INPUT A
5050 POKE F,A
5060 PRINT PEEK F,CHR$ PEEK F
5070 IF PEEK (F+1)=118 THEN STOP
5080 GOTO 5010
```

By now, you probably feel a little as I did when I started studying the violin--the teacher spent what seemed like hours explaining how to hold the violin properly, when all I wanted to do was make music. I'm sorry about that. But we have one more digression to make before we really get into the business of coding.

If you already know all about hexadecimal notation (referred to indiscriminately as "hexadecimal" and "hex" throughout this series), or if you don't know, but don't mind sitting around trying to look wise when you talk with your computing friends or read a book on computing, then you can skip the first part of this chapter. Right up to the section on 256-imal. When you need to know the hex equivalent for a decimal number, or vice versa, you can always look up the conversion in the ZX81 handbook. Or compute your own conversions with a program like the one Tom Woods presented in Vol. 2 No. 2 of this publication.

But the section on 256-imal is important. Read and digest it.

FIG 5-2. THE 1 REM STATEMENT

```
16514 42 E LD HL,
16515 10 GR S (16394)
16516 64 RND
16517 235 FOR EX DE,HL
16518 33 5 LD HL,
16519 125 ? 16509
16520 64 RND
16521 35 7 INC HL
16522 62 Y LD A,
16523 117 ? 117
16524 60 W INC A
16525 190 D CP(HL)
16526 32 4 JR NZ,
16527 249 RAND -7
16528 35 7 INC HL
16529 190 D CP(HL)
16530 200 COS RET Z
16531 122 ? LD A,D
16532 190 D CP(HL)
16533 32 4 JR NZ,
16534 242 PAUSE -14
16535 35 7 INC HL
16536 123 ? LD A,E
16537 190 D CP(HL)
16538 32 4 JR NZ,
16539 237 GOSUB -19
16540 43 F DEC HL
16541 229 FAST PUSH HL
16542 193 AT POP BC
16543 201 TAN RET
```

What is Hexadecimal?

You may already be familiar with binary counting. You're certainly familiar with decimal, although you may not have stopped to think how it works. (As Tom Lehrer points out, tongue in cheek, the important thing is to know how the system works--it doesn't matter whether you get the right answer or not.) Just to review, the meanings of the digits of the two systems are as shown in Figures 5-3 and 5-4. By logical extension, hexadecimal, based on 16, works as shown in Figure 5-5.

FIG 5-3. DECIMAL NOTATION

1325 IN DECIMAL =

```
1 * 10**3 (=1000)
+ 3 * 10**2 (= 300)
+ 2 * 10**1 (= 20)
+ 5 * 10**0 (= 5)

(=1325)
```

But, just as binary requires only two symbols (0 and 1) and decimal requires 10 of them (0-9), hex will require 16 separate characters. We already have 0-9; the other six are taken from the alphabet: A=10, B=11, C=12, D=13, E=14, and F=15. So A9 would be $(10*16+9)=169$; 9A would be

FIG 5-4. BINARY NOTATION

1011 IN BINARY =

1 * 2**3 (= 8d)
+ 0 * 2**2 (= 0d)
+ 1 * 2**1 (= 2d)
+ 1 * 2**0 (= 1d)

(=11d)

(9*16+10)=154; and so on. And C9, which you'll soon learn from constant repetition, is our old friend 201, the code for RETURN.

FIG 5-5. HEX NOTATION

1A2F IN HEX =

1 * 16**3 (=4096d)
+ 10 * 16**2 (= 320d)
+ 2 * 16**1 (= 32d)
+ 15 * 16**0 (= 15d)

(=4463d)

But Why Use Hex?

To be honest, the most important reason to learn hexadecimal is that "everybody does it." Most published listings of the Z80 chip, for instance, are given in hex. So are large numbers of published programs. Typographically, the use of hex makes for a neater looking page--the fact that each hex byte is expressed in exactly two bytes makes it easier to list a program in a symmetrical fashion. And loader programs for hex are a bit simpler than those expressed in decimal.

However, I never accepted "everybody does it" as an excuse from my kids. And this is, after all, a series for beginners. So we'll be using decimal notation primarily. If there's any chance for confusion, I'll follow the accepted convention of adding an "h" or a "d" after a number, such as:

20h=32d (i.e. 20 hex = 32 dec)

or 20d=14h

One other thing. Hex numbers do not contain the letter "O." Anything that looks like an "O" is bound to be a "0" (zero). On the other hand, any number that is not made up of an even number of digits (usually 2 or 4) can not be hex.

Two-Fifty-Six-imal

That sounds like a made-up word, doesn't it? It should--it's my own invention. There's probably a fancy Greek or Latin derivative to describe this form of notation--or more likely a word with one parent from each language, as is the case with "hexadecimal"--but I don't know what it would be. I suppose, by analogy with the abbreviation "hex", it would be called "toof", but I'll let that one go.

Two-fifty-six-imal isn't strictly analogous to decimal and hex, by the way, but is perhaps more like the Binary Coded Decimal notation that you run across from time to time. If it really were analogous to decimal and hex, it would require 256 characters, which would exhaust the Latin, Greek, Hebrew, and Cyrillic alphabets, and then some. Not to mention exhausting our patience.

In the 256-imal system, as used in this series, a number mn usually has the value of $(256*m)+n$. That is, 256 times the value of the first number, plus the value of the second number. You may recognize this system from the earlier chapter, where BC was $256*B+C$. The first number in this case is referred to as the "most significant number", since it has 256 times the weight of the second ("least significant") number.

You can't always count on the most significant number coming first, however. To look ahead a bit, the registers of the Z80 are always NAMED with the most significant byte first. You can remember this by the fact that the HL register originally meant "High-Low." However, the memory usually STORES information with the least significant byte first. So the contents of the memory pair 16388/16389 would be

PEEK 16388 + 256*PEEK 16389.

Don't ask me why Sir Clive did it that way: he probably wasn't the first to do it.

And to make matters worse, there's an exception to the exception--but we'll get to that in due course.

For the moment, remember that the value of a 256-imal number will always be $256*MSN+LSN$, where MSN and LSN stand for Most Significant Number and Least Significant Number, respectively. This fact will never change--the only thing that may change from time to time is whether we print MSN or LSN first.

FIG 5-6. SOME ZX81 ADDRESSES IN 256-IMAL

NUMBER IN DEC	NUMBER IN 256-IMAL		ZX81 SIGNIFICANCE OF THIS ADDRESS
	LSN	MSN	
16388	4	64	RAMTOP
16389	5	64	
16396	12	64	D-FILE
16397	13	64	
16400	16	64	VARs
16401	17	64	
16404	20	64	E-LINE
16405	21	64	
16421	37	64	LAST KEY PRESSED
16422	38	64	
16507	123	64	SPARE BYTES OF MEMORY
16508	124	64	
16509	125	64	FIRST BYTE OF PROGRAM
16514	130	64	FIRST BYTE OF 1 REM

You will see why we need a system like 256-imal if you recall that each byte in computer memory will handle numbers from 0 to 255. Any number higher than 255 is stored in two successive bytes, in 256-imal.

A typical use of 256-imal would be in storing the values of memory addresses. The addresses that you most often will use are from 16507 on up. Figure 5-6 lists a number of ZX81 addresses that are particularly useful, along with their equivalents in 256-ima. You'll notice that you will be using 64 as the most significant number more often than any other value. For this reason, you might find it useful to memorize the fact that $256 \times 64 = 16384$, so that you can quickly compute in your head the 256-imal value of the most-often used memory addresses.

Problem

What is the largest number that can be expressed in 256-imal? See if you can figure it out for yourself before you read further.

Answer

Remember that the largest number a single byte can store is 255. The largest number in 256-imal, then, is two bytes of 255 each, or $256 \times 255 + 255$, or 65535.

You may recognize this as the equivalent of $256 \times 256 - 1$ (or $256^{**}2 - 1$). This gives the 256-imal system a total capacity of 256×256 numbers (counting 0), just as two digits of binary have a repertoire of 4 values (0-3), while two decimal digits can represent any one of 100 values (0-99).

And that's it for now. There will be more.

OFF THE WALL

How Many Combinations?

How big is 64K, really? Have you ever wondered exactly how many possible combinations there are in 65536 memory locations, each capable of assuming values from 0 to 255? Well, 64K bytes is 524288 bits, so the number of possibilities is $2^{**}524288$, or about $2.6E157826$ (26 followed by 524,287 zeros). Printing this number in decimal would take 225 screens on your computer display.

While we can express such numbers, they are obviously far beyond our power to comprehend. So let's break the problem down to a "manageable" magnitude, and just consider a paltry eight bytes. In other words, how many different custom characters can be generated on the Spectrum/2068? It won't take your computer long to tell you that $2^{**}64 = 1.84E19$. There, that's better. Or is it? You could write a simple program (see Paul Bingham's bit generator for starters) consisting of eight nested loops, each running from 0 to 255. If the 2068 can generate 10 such characters per second (actually pushing it a little) how long will it take to run the program?

Well, I'll tell you: just a little over 58 BILLION YEARS. If you printed each character on the TS2040, 32 characters per line, the resulting printout would circumscribe the known solar system (orbit of Pluto) 50 times. Astronomers - feel free to check my figures.

-fn

Classified !

Do you have something to sell? Let everyone know. Just \$2.75 a line will put it here. 32 column (screen) format, include spaces in your text and send CK. or MO with your listing.

Send to SWN Classified, 9016 Flicker Pl., Columbia, MD 21045

Get cash for that unused board or addon! SELL IT!!

FOR SALE: 2040 printer, \$40; 2068 w/3 cartridges-Casino-Vu3d-Vufile, \$135; 2050 Modem+source, \$140; +more software. Ian Singer, 30 Brookmount Rd., Toronto, Ont., Canada M4L 3N1

Teachers: 255 classes, entire pupil or class file on screen in 1 sec., weight avgs., bonus/-demerit pts., and more! High 16K cap. from 279 pupils- 5 grades to 64 pupils- 66 grades. 5x more on 2068! Money back guarantee! TS Grader 1000 = \$15.95 or \$19.95 on 2068. \$1.00 refundable on order for info sheet. Robert Fischer, 221 Scoggins St., Summerville, GA 30747

Wanted: Memotech RS232 i/f and cable for TS1000. Will pay \$50.00. Anthony Oresteen, 452 Orion, Batavia, IL 60510, (312) 879-5608.

FOR SALE: TS1000 w/16K+manuals, \$30; ZX81 w/filesixty kbd & manuals, no power supply, \$20; Memotech 64K ram, has split edge connector, \$25; ZX printer +4 rolls of paper- works but goes off by itself, \$20. All above + shipping at cost. Several software titles @ \$5 ea. John Tooley, RD#2 Box 120E Milton, DE 19968, 302 856-5260

PRODUCTS ON PARADE

Uploader 2000

E-Z Key
Suite 75, 711 South Artery,
Quincy, MA 02169

Program Type: Utility (Program Conversion)
Machine: TS2068
Price: \$19.95 (Optional Load filter \$9.95)

If you're like me, the TS2068 is not the first computer you have owned. You probably have a ZX81 or TS1000, with plenty of programs to run on it. Now that you've moved on to the more sophisticated TS2068, you probably think that in order to build a library for it, you'll have to start from scratch. Right?

WRONG! Almost any program you have in BASIC for the ZX81/TS1000 can now be used on the TS2068 with the use of UPLOAD2000. This is a program that allows you to convert most BASIC 1000 programs into working versions for the 2068.

Notice I said BASIC program. The program must be written totally in BASIC, with no machine code.

The procedure to translate is quite simple.

- 1) First, you load UPLOAD2000 into your computer.
- 2) Next, you type a RAND USR command given.
- 3) Then load in your TS1000 program.
- 4) After it finishes loading, list the program. It should look exactly like it does in the 1000.
- 5) Next, you will have to edit the program. Remove all FAST and SLOW commands. Other keywords, such as CHR\$, CODE, PEEK and POKE will have to be modified to fit 2068 code. [You will also have to change scaling, etc. for PLOT/UNPLOT, and change any code that uses SCROLL. You might also want to place entire subroutines into multi-statement lines, modify INPUT prompts and commands, and modify legends, etc. to include lower case. -ed.]
- 6) Once you have finished the "mandatory" editing, you can save this new version for use, or you may wish to modify it further. How about adding color and sound?

Depending on the length of the program, editing may be from none to quite a bit. However, I would rather edit a working version than re-key an entire program.

I have used UPLOAD2000 on numerous occasions, and the results have been very good. To insure a good load with your 1000 program, I recommend getting the LOAD FILTER that E-Z KEY has available. It plugs in line with the cord between the computer and the cassette recorder.

I feel that UPLOAD2000 is a very good utility, and recommend it to those who want to make use of their TS1000 BASIC programs in the TS2068.

Reviewed by:
Bill Ferrebee 115 N. 7th Ave. Paden City, WV 26159
River Cities Smart BBS (304) 652-1416

Appointment Watch

WMJ Data Systems
4 Butterfly Drive
Hauppauge, New York 11788

Price: \$10 (TS1000), \$12 (TS2068)

APPOINTMENT WATCH is a spread sheet program written in machine code and BASIC. It requires at least 16K of memory. In 16K, Appointment Watch allows the user to enter and store 100 appointments. It allows you to enter the Date, Time, and Place of your appointments, as well as with whom they are, and the subject of the meeting. Also provided is space to mark whether or not the appointment has been confirmed.

Data entry is done from the main menu with the "Enter A Record" selection. Screen prompts lead the user through filling in the individual fields of a record, and provide an indication as to how long a field may be. If you make a mistake, or if, say, the time of the original appointment is changed, corrections can be made to individual fields. Such commands are available from the spread sheet display. Other main menu selections are View Spread Sheet, Print Out Information, Clear Spread Sheet Of All Data, and Save Info To Tape.

The View Spread Sheet command displays the program title, ten of the data/appointment entries, and menu options. APPOINTMENT WATCH uses a "window" to view the appointment data, via scroll routines invoked with the arrow keys. The up/down scroll advances or regresses the appointment list, and the right/left scroll allows you to view each of the data fields in an appointment line. The scroll routines are written in machine code, making them very quick and truly fascinating to watch.

The Print Out Information selection does just what it says. It works with full size printers equipped with Memotech or Aerco printer I/Fs, as well as with Timex/Sinclair printers. When the Print Out command is invoked, screen prompts lead the user to printing out all the data contained in the spread sheet or just selected items. However, if the user accidentally enters "0", the program will halt, requiring a "GOTO 100" command to restart without losing all the data.

The Clear Spread Sheet Of All Data command can be suicidal as implemented in the program. Selecting this command will IMMEDIATELY cause all data fields to be blanked, sending all of your data off to never-never land, with no chance of recovery. In my house, with kids and cats, all of which seem to take great joy in pressing the computer keyboard keys at the most inopportune times, such commands should always be protected with "Are you SURE you want to do this?". Since the program is written mostly in Basic, this program can easily be user-modified to include such protection.

The Save Info To Tape main menu selection is self explanatory. APPOINTMENT WATCH will allow you to enter a title as a name for subsequent loading. If you just press "ENTER" for the name, or if you press

the BREAK key on the prompt "Press Any Key", the program will stop, again necessitating a "GOTO 100" to restart. No problems were encountered modifying APPOINTMENT WATCH for disc use.

If you've overcome the "Oh, another one of those good-idea-but-impractical-to-use programs" train of thought, and have followed along this far, then read on. The "meat" of this review is yet to come! Like most programs of this genre, APPOINTMENT WATCH probably isn't worth your time and money at face value. What makes it worth \$10 of your hard-earned cash are the MC routines. Dissecting the 395 bytes of machine code that controls the window IS worth your money, time, and effort. The MC program used in

APPOINTMENT WATCH, or your own variations of it, could be used in your own programs. Moving windows, pull down drawers, etc., such as are used by IBM and Apple, can thus be implemented on the Timex/Sinclair.

APPOINTMENT WATCH comes on cassette tape with a brief instruction sheet. Side one of the tape contains a 22 second test program that allows you to check the playback volume control for the proper setting. Side two contains the actual APPOINTMENT WATCH program.

Reviewed by:

Jeffrey Moore 602 S. Mill Street Louisville, Ohio 44641

CUSTOMIZE YOUR M-SCRIPT

Mark Fisher
CATS USER GROUP
700 Erie Ave.
Takoma Park, MD

I am one of those people who can't have a program for long before I find something to improve or otherwise fool with. In this case, the victim is MSCRIPT. MSCRIPT is a beautiful, full featured word processor. In a month of using the program, I've found only four things to complain about.

1. The characters were hard to read, due to the Timex's method of getting 64 characters on a line. This was solved by Jack Dohaney, in his Fat Bits program as listed in issue 2/4 of SWN.
2. The space bar alternately sticks and prints double. This is a Timex hardware problem, due to their 29 cent keyboard.
3. DELETE works opposite to Timex convention, deleting to the right rather than the left.
4. TAB is awkwardly placed, requiring pressing the CAPS SHIFT, SYMBL SHIFT, and 8 keys together. This is the one that can be changed.

Most programs use the ROM keyboard decode subroutine to translate the signals received from the keyboard into the characters represented. MSCRIPT doesn't do this. It does all of its own keyboard decoding, using a set of tables located between 42582 and 42837. Since these tables are in RAM, you can change that table, even so far as creating a Dvorak keyboard, by POKEing new values.

MSCRIPT uses these four tables to translate the electrical key press to an ASCII code for the corresponding letter. Each table is represented by a column in figure #1. It chooses among the four columns on the basis of the status of the CAPS and SYMBL shift keys. The first column headings reflect this. Each of these tables starts at a different address - these addresses are given above each column. MSCRIPT then chooses the offset that corresponds to the key pressed, moves over to the correct column, and returns with the entry at that position.

For example, pressing an unshifted "d" causes MSCRIPT to look in the table starting at 42582, and

return with the number at offset seven. This is 100, the ASCII code for "d". If the same key were depressed while holding both CAPS and SYMBL shifts, a 196 would be returned.

In general, numbers followed by a blank have no effect when pressed. In our example, the 196 shown in the table will be returned, but nothing will happen on the screen. Sometimes that's ok (e.g., we don't want to get a character when we press the CAPS SHIFT), but it is in the interest of speed to move as many functions to Caps Shift keys as possible.

As an example of how to implement these changes, we will move the TAB function key to the shifted one key.

1. LOAD "MSCRIPT" CODE.
2. POKE the TAB value, 152, to the key you wish to use for tab (in this case SHIFT 1).
3. Find the offset for the "1" key (15), and add it to the base address of the CAPS SHIFT column (42646) to get the correct address: 42661.
4. Type POKE 42661,152, and ENTER, to put the TAB function on the shifted 1.
5. SAVE "MSCRIPT" CODE,(start,length)

Other often used functions are INSERT (CAPS, SYMBL, I) and MERGE (CAPS, SYMBL, M). These could be placed on the unused CAPS shifted 3, 4, or 9. While we're at it, how about making the unshifted 5, 6, 7, and 8 act as the cursor controls? This would allow easy movement around the text, while still allowing the numbers to be printed if they were CAPS shifted. This would be accomplished by exchanging the values at offsets 19, 22,23, and 24 between the No shift and CAPS shift columns.

BACKGAMMON

Looking for a real challenge? Tired of slow response time? Practicing for a tournament? If so, you'll want to try our Backgammon game. Features: full-screen graphics, break disabled, 100% machine code, clean loading, smart play, top rating from TSUG review, multi-game scoring, cube doubling, simple operation, documentation included. Requires ZX81 or TS1000 w/16K RAM. Available on cassette for only \$10. In a 10 game test against PSION, BLOCAL won every game! Write for our FREE catalog listing 100 programs for TS 1000 and 2068. Please include expir. date on MC/VISA orders.



BLOCAL
SOFTWARE, INC.
P.O. Box 965

Stinson Beach, CA 94970

MSCRIP Keyboard Decode Table

Base Address	No Shift 42582	CAPS shift 42646	SYMBL shift 42710	CAPS+SYMBL shift 42774	Offset in table
	00	00	00	00 (caps shift)	0
122	z	90 Z	58 :	218	1
120	x	88 X	96 +	216	2
99	c	67 C	63 ?	195 copy block	3
118	v	86 V	47 /	214	4
97	a	65 A	00	193	5
115	s	83 S	00	211 sub-delete	6
100	d	68 D	92	196	7
102	f	70 F	00	198	8
103	g	71 G	00	199 printer code	9
113	q	81 Q	00	209 restore line	10
119	w	87 W	00	215	11
101	e	69 E	00	197 cursor to END	12
114	r	82 R	60 <	210 remove block	13
116	t	84 T	62 >	212 cursor to TOP	14
49	1	129	33 !	145	15
50	2	130 capLK	64 @	146	16
51	3	131	35 #	147	17
52	4	132	36 \$	148	18
53	5	133 lft	37 %	149 cur full left	19
48	0	8 del	95	24	20
57	9	137	41)	153	21
56	8	136 rgt	40 (	152 tab	22
55	7	135 up	39 '	151 page up	23
54	6	134 dwn	38 &	150 page down	24
112	p	80 P	34 "	208 PRINT menu sel	25
111	o	79 O	59 ;	207	26
105	i	73 I	00	201 insert text	27
117	u	85 U	93]	213 unmark block	28
121	y	89 Y	91 [	217	29
13	ent	13 ent	13 ent	13 ent	30
108	l	76 L	61 =	204	31
107	k	75 K	43 +	203	32
106	j	74 J	45 -	202	33
104	h	72 H	94 s	200 HELP summary	34
32	sp	128 menu	128 menu	128 menu	35
00		00	00	00 (smb shft key)	36
109	m	77 M	46 .	205 merge text blk	37
110	n	78 N	44 ,	206 new page	38
98	b	66 B	42 *	194 set block mark	39

TS1000 MOVE IT !

Fred Nachbaur

Here is the second of a short series of intermediate-level machine-code articles, for those of you who enjoyed playing with the CLS routines of issue 2:4. Unlike the last installment, however, the routine here is actually useful, not "just another pretty screen!"

One of the nice features of the Z80 is that it has a number of commands which actually are a whole subroutine in a single op-code or machine-language instruction. Quite probably the most powerful of these are the "block-transfer" group of commands. These allow you to easily transfer blocks of memory anywhere you wish, at a rapid rate; 16K can be transferred in about .1 sec. in FAST mode and on the Spectrum/2068, and in about 1/2 sec. in ZX81 SLOW mode. The LDIR (load-increment-repeat) command can be used for filling from the bottom up, and LDDR (load-decrement-repeat) goes the other way (allowing for data-block overlap either way). There are also "semi-automatic" commands (LDI, LDD) which allow you

to jump out of the loop if you need to. Similarly, string searches and other jobs requiring comparisons go a lot easier using CPIR (compare-increment-repeat), CPDR, CPI and CPD commands. There are even similar "routines in an op-code" for input/output use (INIR, OTIR, etc.)

These commands all have one thing in common; simplicity of use. For example, the LDI/LDD group of commands only require that you:

- 1) Fill the BC register with the number of bytes to transfer,
- 2) Fill the DE register with the destination address (where you want it), and
- 3) Fill the HL register with the source address (where it is now).
- 4) Then LDIR or LDDR, and the move takes place.

The source block is left unchanged, unless of course the source and destination blocks overlap.

I've mentioned one use for this before, in this magazine and elsewhere; storing multiple programs in RAM and calling any at will. This is especially useful for ZX81's with 64K, since the vast majority of software for the machine is designed for 16K. You could thus keep two full-16K programs in high memory (8000-BFFF and C000-FFFF), and bring either down into the operating area (4000-7FFF or 16-32K). See Bob Hartung's article in this issue for further information on these commands. MANAGE UP TO "TROIS"

But imagine for a moment if you had a command, like LDIR, that SWAPS or exchanges two blocks of memory. You could have THREE separate programs available at all times! You can swap using LDIR or LDDR, of course, but the larger the block is the more memory you tie up for the temporary storage. To continue the same example, you still could only store two 16K programs at a time in 64K. Well, the short routine of Listing 1 gets around this. In addition to its usefulness, it might help you understand how the block transfer commands actually operate. It is analogous to LDIR; as with LDIR, you load BC with the number of bytes, and DE and HL with the start addresses of the blocks to swap. I call it "SWIR," to reflect this similarity with the "canned" Z80 commands. Note that there would not be much point in having a separate "SWDR" routine to go the other direction, since you'd normally use this only on non-overlapping blocks. For a brain teaser, though, figure out what happens if the blocks do overlap, how to modify the routine to SWDR, and what this would do with overlapping blocks.

Note also that although this routine is comparatively slow (takes about twice as long as two uses of LDIR), it still swaps 16K in about 1/2 second in FAST mode. Another way of looking at it is that it will swap the contents of the ZX81 display file in about .1 sec. in SLOW mode.

So what can you do with it? Well, Listing 2 shows a couple short drivers to swap programs as mentioned above. You'll have to use FIL1 and FIL2 to store two of the three programs you'll be running. This is because the high-memory region will contain "garbage" or all zeros on power-up, and swapping this into the operating region will cause a crash. So here's what you do:

- 1) Load the first program, and call FIL1 (RAND USR 8207).
- 2) Then load the second program and call FIL2 (RAND USR 8222).
- 3) Now load the third program.
- 4) Henceforth, instead of using NEW, use SWP1 (RAND USR 8238) and SWP2 (RAND USR 8243) to exchange the current program with the one stored at slot 1 (32-48K) or slot 2 (48-64K) respectively.

No matter how many times you call SWP1 and SWP2, you'll always have all three 16K programs at your disposal. (Example: your filing program, HotZ II, and a program you're working on.) Note that all three programs must have the same RAMTOP setting, or you'll have stack problems after a swap. Also note that the SWP driver only swaps 3F20 bytes instead of the full 4000h (16K). This is to prevent overwriting the stack, while leaving ample room above E LINE for most "16K" programs. I chose 3F20 because HotZ II's program area ends at 7F20 (4000+3F20). If you're dealing with REALLY cram-packed 16K programs, you might have to raise this somewhat. PRINT PEEK 16404+256*PEEK 16405 to get E LINE of each program. Also PRINT PEEK 16386+256*PEEK 16387 to get ERR SP. The value you place into 203C-203D (8252-8253d) should be higher than the highest E LINE, but at least 40 bytes lower than the lowest ERR SP.

LISTING 1 SWIR SUBROUTINE

```

0000 007E SWIR LD A,(HL) : GET CHARACTER FROM BLOCK 1
0001 0080 PUSH AF : SAVE IT
0002 0082 LD A,(DE) : GET CHARACTER FROM BLOCK 2
0003 0084 LD (HL),A : PUT IT INTO BLOCK 1
0004 0086 POP AF : GET CHAR. FROM BLOCK 1
0005 0088 LD (DE),A : AND PUT IT INTO BLOCK 2
0006 008A INC HL : NEXT BLOCK 1 LOCATION
0007 008C INC DE : NEXT BLOCK 2 LOCATION
0008 008E DEC BC : DECREMENT COUNTER
0009 0090 LD A,B : CHECK HIGH BYTE
000A 0092 OR C : AND LOW BYTE
000B 0094 RET Z : ALL DONE IF BOTH ARE ZERO
000C 0096 JR SWIR : ELSE CONTINUE WITH NEXT CHAR.

```

LISTING 2 3-PROGRAM MANAGER

```

000F CD230F FIL1 CALL 0F23 : CALL FAST MODE
0010 010040 LD BC,4000 : NO. BYTES=16384
0011 110080 LD DE,8000 : DESTINATION=32768 (32K=SLOT1)
0012 210040 LD HL,4000 : SOURCE=16384
0013 EDB0 LDIR : FILE PROGRAM IN SLOT 1
0014 00 RET : RETURN
0015 CD230F FIL2 CALL 0F23 : CALL FAST MODE
0016 010040 LD BC,4000 : NO. BYTES=16384
0017 1100C0 LD DE,C000 : DESTINATION=49152 (48K=SLOT2)
0018 210040 LD HL,4000 : SOURCE=16384
0019 EDB0 LDIR : FILE PROGRAM IN SLOT 2
0020 00 RET : RETURN
0021 00 NOP
0022 110080 SWP1 LD DE,8000 : SWAP WITH SLOT 1
0023 1803 JR SWAP :
0024 1100C0 SWP2 LD DE,C000 : SWAP WITH SLOT 2
0025 05 SWAP PUSH DE : SAVE SLOT LOCATION
0026 CD230F CALL 0F23 : CALL FAST MODE
0027 01 POP DE : GET SLOT LOCATION
0028 01203F LD BC,3F20 : NO. BYTES=16160
0029 210040 LD HL,4000 : EXCHANGE LOCATION=16384
0030 CD0020 CALL SWIR : SWAP CURRENT PROGRAM WITH SEL
0031 00 RET : ECTED SLOT, AND RETURN.

```

A.F.R. SOFTWARE

PRESENTS:

THREE EXTRAORDINARY PROGRAMS FOR THE
"Timex-Pinclair"
 Computer

"ZX/CALC"- CALCULATOR
 "ZX/TEXT"- WORD PROCESSOR
 "ZX/CALENDAR"- APPOINTMENT CALENDAR

FOR COMPLETE DETAILS
 WRITE:

A.F.R. SOFTWARE
 1605 Pennsylvania Ave.
 # 204
 Miami Beach, Florida
 33139

Other Applications

For the 2068 (or ZX81 for that matter), you could use this to swap display files (Bob Hartung's article), swap sets of variables (such as different loads of a file program), even different programs that share the same variables set. Your job will only be determining the source, destination, and number of bytes. Frequently some of this information will be contained in system variables, like D_FILE, VARS, etc.

I used it to create an alternate display file in the middle of the program area which could contain data and must not be overwritten. The problem was that you can't move the display file into the top 16K of high memory directly, so I used SWIR to help out. Here's how it works: SWIR exchanges the data in the "alternate display file" area with a screen located

in hi-mem, then the D_FILE pointer is moved to the new location. When done, D_FILE is loaded with the original value, and SWIR is called again to restore the positions of the auxiliary screen and the data. There's no reason you couldn't take this further and have several video pages in hi-mem, swapping them about at will with a POKE (which slot) followed by a USR call (swap). This would be especially useful for the 2068, with its 6K+ display file.

How about using it within a program to swap two pieces of software that use the same region of memory? Often this is a lot easier than relocating tricky software packages. Simply use the swap routine to boot whichever chunk of code is needed at any given time.

I'm sure you can find applications of your own. Next time we'll take a quick look over a simple check-sum routine, useful for verifying tape loads.

2068 PRINT COMMAND COMPILER

William F. Powers
MAXSOFT
611 Franklin St.
Hamilton, OH 45013

Machine code (MC) programming is fantastic for writing programs that move fast, and is essential for programs that must move or sort a large amount of data or move it in a way that causes smooth action on the screen, such as a moving graphics game.

Before I lose the readers who would like to program in machine code, but have been afraid to try, let me say two things:

- 1) if you can start with an idea and turn it into a BASIC program, YOU CAN LEARN TO PROGRAM IN M/C, and
- 2) if you are leary of trying, you can rest assured that there is no program that you can enter into your computer that will cause a permanent problem. ALL program crashes can be "fixed" with the power switch or reset button, so don't be afraid to experiment.

Assembly language is the ideal method for programming MC on the Timex, but some other languages allow you to write using BASIC-like words, which are then converted to machine-code before running. This lets you work in a high level language, but the program runs with the faster speed of the lower level language. (Such is the case with BASIC compilers, like the one available from JRC Software, which I have used with good results, and languages such as PASCAL and FORTRAN.)

No matter which way you decide to go (and it is quite acceptable to mix the methods), it is beneficial to know how to use ROM calls. These are routines in the computer's BASIC operating system which you can use from within your own program. As a result, you don't "re-invent the wheel." For those TS2068 owners who are interested, but have not noticed the enticing menu of ROM call "goodies" in SyncWare News Vol. 2, No. 1, I call them to your attention and recommend that you study them.

This article addresses itself to one of the calls in particular; the PRINT routine at 2159h (8537d). The author, Ray Kingsley, points out the fact that you

can put the message in memory, point the system variable CH_ADD to the location of the start of the message, and call the PRINT routine. He also very helpfully mentions that if you can get the floating point form of the numbers that you use (e.g. AT 3,5; or FLASH 1; etc.) into the memory area in the same form as it is in the PRINT statement, then the AT, TAB, INK, PAPER, etc. instructions will work as well.

Note also that BASIC variables will work (e.g. TAB x) if the variable is defined before the PRINT routine is called. Also, since all of the instructions associated with PRINT use integer numbers less than 255, you can use a command like PRINT FLASH PEEK 40000. You must pre-set location 40000 (or wherever you choose) from inside your MC program before calling the PRINT routine.

The following program will allow you to enter PRINT statements in any line from 1 to 999, one PRINT statement per line. RUN 1000 will then compile your PRINT statements into machine code using a ROM call. The program will first ask you to where you want the code located, and then place into memory an 18-byte driver routine (explained later). This is followed by the message AT/TAB/PAPER etc., and then a report of the memory location to call for the PRINT work which was contained in the line. The line number of the source line is also displayed.

Listing 1.

```
1000 LET get=PEEK 23635+256*PEEK
23636: REM prog
1010 INPUT "ENTER start address.
";put
1020 IF PEEK (get+4)<>245 THEN P
PRINT "The last address used was
";put-1: STOP
1030 PRINT " LINE ";(256*PEEK ge
t+PEEK (get+1));"starts AT ";put
1040 LET p1=put+18: DATA 42,93,9
2,229,33,p1-(INT (p1/256)*256),I
NT (p1/256),34,93,92,205,89,33,2
25,34,93,92,201
1050 RESTORE 1040: FOR r=1 TO 18
: READ x: POKE put,x: LET put=pu
t+1: NEXT r
1060 LET get=get+5
1070 IF PEEK get=14 THEN FOR r=1
TO 6: POKE put,PEEK get: LET ge
t=get+1: LET put=put+1: NEXT r
1080 IF PEEK get=13 THEN POKE pu
t,PEEK get: LET put=put+1: LET g
et=get+1: GO TO 1020
1090 POKE put,PEEK get: LET put=
put+1: LET get=get+1: GO TO 1070
```

The routine is shown in listing 1, and operates as follows:

- 1000 - Gets the start of the program area where the PRINT statements are located
- 1010 - Tell the computer where to put the code
- 1020 - If the first token word is not a PRINT (CHR\$ 245) then stop the program.
- 1030 - Display the line # and the memory location to call.
- 1040 - Set p1 to the current value of PUT+18 (just past the driver routine) and also the DATA for setting up the driver. Notice that formulas and variables are legal in a DATA statement.
- 1050 - Start DATA at line 1040, POKE the 18 bytes of machine code.
- 1060 - Jump over the bytes with the line # (2), line length (1), and the PRINT (1 byte).
- 1070 - If the "get" byte is the numbers slug, then POKE the next 6 bytes.
- 1080 - If the "get" byte is the END OF LINE (ENTER) character, then POKE the ENTER and start the next BASIC line.
- 1090 - If not, POKE the "get" byte and go look at the next one.

The 18 bytes of the driver for each PRINT statement are much like Mr. Kingsley suggested, except it starts by getting the current value of CH_ADD (in this case 60018 if the driver code starts at 60000), call the PRINT routine, put the old CH_ADD value back and then return.

In order to preserve the original CH_ADD value as this system does, your machine code must CALL the address that is shown on the screen, instead of JPing to it. The CH_ADD variable may be changed to any value you like as long as you stay in machine code, but MUST be restored to its initial value if your program ever returns to BASIC.

```

ADDR WRITE LABEL MNEMONIC
=====
EA60 2A535C LD HL, (prog)
EA63 E5 PUSH HL
EA64 2172EA LD HL, EA72
EA67 0D5921 CALL 2159
EA6A E1 POP HL
EA6B 22535C LD (prog), HL
EA6E C9 RET

```

```

ADDR HEX DEC ADDR CHR$
-----
EA60 2A 42 60000 *
EA61 53 83 60001 S
EA62 5C 92 60002 \
EA63 E5 229 60003 RESTORE
EA64 21 33 60004 !
EA65 72 114 60005 r
EA66 EA 234 60006 REM
EA67 0D 205 60007 STEP
EA68 59 89 60008 Y
EA69 21 33 60009 !
EA6A E1 225 60010 LLIST
EA6B 22 34 60011 "
EA6C 53 83 60012 S
EA6D 5C 92 60013 \
EA6E C9 201 60014 <>

```

SyncWare News

CONTENTS

For Your Support....	3
Forum....	4
2068 Cassette Connection....	7
Nachbaur	
Run ZX81 Machine Code in High Memory	10
Oliger, Moore	
Quick Screen Display 2068....	13
Hartung	
EPROM Programmer Part 2....	14
Oliger	
Basil's Compendium....	16
Wentworth	
Off the Wall....	18
Nachbaur	
Classified Ads....	18
Uploader 2000 Review	19
Ferrebee	
Appointment Watch Review....	19
Moore	
Customize Your Mscript....	20
Fisher	
TS1000 Move It!....	21
Nachbaur	
2068 Print Command Compiler....	23
Powers	

FIRST-CLASS MAIL
U.S. POSTAGE
PAID
Jefferson, NH
Permit No. 1

Submissions, announcements, letters to the editor, and all other material of editorial interest should be sent to the Maryland address.

Copyright 1985 SyncWare News

Editor:
Tom Bent
9016 Flicker Place
Columbia, MD 21045

Technical Director:
Fred Nachbaur
902 Hoover St.
Nelson, BC Canada V1L-4X6

Advertising, Subscriptions:
Thomas B. Woods
P.O. Box 64
Jefferson, NH 03583

Subscribe ! \$ 16.95

1 year (6 issues)

Add \$3 a year in Canada; all other foreign add \$5 per year.